CHAPTER 3 DEBUG MONIT

3.8 Data Conversion (DC) ....
3.9 Dump S-Records (DU) ....
3.10 Go Direct (GD) ....
3.11 Go To Next Instruction (GN)......................
3.12 Go Execute User Program (GO).................
3.13 Go To Temporary Breakpoint (GT)............
3.14 Help (H) ....
3.15 Load S-Records From Host (LO) ....
3.16 Macro Define/Display/Delete (MAL/NOM
3.17 Macro Edit (MAE).......................................
3.18 Macro Expansion Listing Enable/Disable (
3.19 Memory Display (MD) ....
3.20 Memory Modify (MM)..............................
3.21 Memory Set (MS) ....
3.22 Offset Registers Display/Modify (OF) ....
3.23 Printer Attach/Detach (PA/NOPA).............
3.24 Port Format (PF) ....
3.24.1 List Current Port Assignments ....
3.24.2 Port Configuration ....
3.24.3 Port Format Parameters ....
3.24.4 New Port Assignment........................
3.25 Register Display (RD) ....
3.26 Cold/Warm Reset (RESET)........................
3.27 Register Modify (RM) ....
3.28 Register Set (RS) ....
3.29 Switch Directories (SD).............................
3.30 Trace (T) ....
3.31 Trace On Change Of Control Flow (TC)....
3.32 Transparent Mode (TM) ....
3.33 Trace To Temporary Breakpoint (TT)........
3.34 Verify S-Records Against Memory (VE)...

CHAPTER 4 ASSEMB

4.1 Introduction...
4.1.1 M68300 Family Assembly Languag
4.1.1.1 Machine-Instruction Operation
4.1.1.2 Directives................................
4.1.2 M68300 Family Resident Structured
4.2 Source Program Coding.............................
4.2.1 Source Line Format ....
4.2.1.1 Operation Field........................
4.2.1.2 Operand Field ....

M68CPU32BUG D

USER'S M

M68CPU

TABLE OF C

CHAPTER 1 GENER

1.1 Introduction.......
1.2 General Description.......
1.3 Using This Manual.......
1.4 Installation and Start-Up.......
1.5 System Restart.......
1.5.1 Reset.......
1.5.2 Abort.......
1.5.3 Break.......
1.6 Memory Requirements.......
1.7 Terminal Input/Output Control.......

CHAPTER 2 DEBUG MO

2.1 Introduction.......
2.2 Entering Debugger Command Lines.......
2.2.1 Syntactic Variables.......
2.2.1.1 Expression as a Parameter.......
2.2.1.2 Address as a Parameter.......
2.2.1.3 Offset Registers.......
2.2.2 Port Numbers.......
2.3 Entering And Debugging Programs.......
2.4 Calling System Utilities From User Progra
2.5 Preserving Debugger Operating Environme
2.5.1 CPU32Bug Vector Table and Works
2.5.2 CPU32Bug Exception Vectors.......
2.5.2.1 Using CPU32Bug Target Vecto
2.5.2.2 Creating Vector Tables.......
2.5.2.3 CPU32Bug Generalized Excep
2.6 Function Code Support.......

CHAPTER 3 DEBUG M

3.1 Introduction.......
3.2 Block Of Memory Compare (BC).......
3.3 Block Of Memory Fill (BF).......
3.4 Block Of Memory Move (BM).......
3.5 Breakpoint Insert/Delete (BR/NOBR).......
3.6 Block Of Memory Search (BS).......
3.7 Block Of Memory Verify (BV).......

LIST OF

FIGURES

1-1. CPU32Bug Operation Mode Flow Diagram
1-2. BCC Memory Map ....

LIST OF

TABLES

2-1. Debugger Address Parameter Format.........
2-2. CPU32Bug Exception Vectors ....
3-1. Debug Monitor Commands ....
4-1. CPU32Bug Assembler Addressing Modes.
5-1. CPU32Bug System Call Routines ....
6-1. MCU CPU Diagnostic Tests........................
6-2. Memory Diagnostic Tests...........................
B-1. Self-Test Error Messages...........................
C-1. CPU32Bug Customization Area.................
C-2. MCU SCI Communication Formats ....
C-3. Rev. A Chip Selection Summary................
C-4. Rev. B Chip Selection Summary................
C-5. BCC Rev. C Chip Selection Summary.......
C-6. PFB Rev. C Compatibility..........................

CHAPTER 4 ASSEMBLER/D

4.2.1.3 Disassembled Source Line ......
4.2.1.4 Mnemonics and Delimiters .....
4.2.1.5 Character Set ....
4.2.2 Addressing Modes............................
4.2.3 Define Constant Directive (DC.W)..
4.2.4 System Call Directive (SYSCALL) .
4.3 Entering and Modifying Source Program...
4.3.1 Executing the Assembler/Disassemb
4.3.2 Entering a Source Line ....
4.3.3 Entering Branch and Jump Addresse
4.3.4 Assembler Output/Program Listings ....

CHAPTER 5 SY

5.1 Introduction...
5.1.1 Executing System Calls Through TR
5.1.2 Input/Output String Formats.............
5.2 System Call Routines.................................
5.2.1 Calculate BCD Equivalent Specified
5.2.2 Parse Value, Assign to Variable (.CH
5.2.3 Check for Break (.CHKBRK) ....
5.2.4 Timer Delay Function (.DELAY).....
5.2.5 Unsigned 32 x 32 Bit Divide (.DIVU
5.2.6 Erase Line (.ERASLN).....................
5.2.7 Input Character Routine (.INCHR) ..
5.2.8 Input Line Routine (.INLN)..............
5.2.9 Input Serial Port Status (.INSTAT)..
5.2.10 Unsigned 32 x 32 Bit Multiply (.MU
5.2.11 Output Character Routine (.OUTCH
5.2.12 Output String Using Pointers (.OUT
5.2.13 Print <CR><LF> (.PCRLF) ....
5.2.14 Read Line to Fixed-Length Buffer (.
5.2.15 Read String Into Variable-Length Bu
5.2.16 Return to CPU32Bug (.RETURN)...
5.2.17 Send Break (.SNDBRK)...................
5.2.18 Compare Two Strings (.STRCMP) ..
5.2.19 Timer Initialization (.TM_INI).........
5.2.20 Read Timer (.TM_RD).....................
5.2.21 Start Timer at T=0 (.TM_STR0) ......
5.2.22 Output String with Data (.WRITD/W
5.2.23 Output String Using Character Coun

CHAPTER 6 DIAGNOST

6.1 Introduction..
6.2 Diagnostic Monitor....................................
6.2.1 Monitor Start-Up.............................
6.2.2 Command Entry and Directories......
6.2.3 Help (HE)..
6.2.4 Self Test (ST)..................................
6.2.5 Switch Directories (SD)...................
6.2.6 Loop-On-Error Mode (LE)...............
6.2.7 Stop-On-Error Mode (SE)................
6.2.8 Loop-Continue Mode (LC)...............
6.2.9 Non-Verbose Mode (NV).................
6.2.10 Display Error Counters (DE)...........
6.2.11 Clear (Zero) Error Counters (ZE).....
6.2.12 Display Pass Count (DP)..................
6.2.13 Zero Pass Count (ZP).......................
6.3 Utilities..
6.3.1 Write Loop......................................
6.3.2 Read Loop.......................................
6.3.3 Write/Read Loop.............................
6.4 CPU Tests For The MCU (CPU)................
6.4.1 Register Test (CPU A).....................
6.4.2 Instruction Test (CPU B)..................
6.4.3 Address Mode Test (CPU C)...........
6.4.4 Exception Processing Test (CPU D)
6.5 Memory Tests (MT)...................................
6.5.1 Set Function Code (MT A)..............
6.5.2 Set Start Address (MT B).................
6.5.3 Set Stop Address (MT C).................
6.5.4 Set Bus Data Width (MT D).............
6.5.5 March Address Test (MT E)............
6.5.6 Walk a Bit Test (MT F)...................
6.5.7 Refresh Test (MT G)........................
6.5.8 Random Byte Test (MT H)..............
6.5.9 Program Test (MT I)........................
6.5.10 Test and Set Test (MT J)..................
6.6 Bus Error Test (BERR)..............................

APPENDIX A S-RECO

A.1 Introduction..
A.2 S-Record Content.......................................
A.3 S-Record Types..
A.4 S-Records Creation....................................

APPENDIX B SELF-TE

B.1 Introduction..

APPENDIX C USER CU

C.1 Introduction..
C.2 CPU32BUG Customization.........................
C.3 Customization Table..................................
C.4 Communication Formats............................
C.5 BCC REV. A Chip Selection Summary.....
C.6 BCC REV. B Chip Selection Summary......
C.7 BCC REV. C Chip Selection Summary.....
C.8 Platform Board (PFB) REV. C Compatibili
C.9 CPU32BUG Questions and Answers.........

M68CPU32BUG/D REV 1

NO

In order for high-baud rate CPU32Bug and the terminal to must use XON/XOFF handshaki have missing characters, check th handshaking is enabled.

3. Power up the system. CPU32Bug message (which includes version num

1.5 SYSTEM RESTART

There are three ways to initialize the system to appropriate system restart technique.

1.5.1 Reset

The M68300PFB platform board reset switch ret switch is first pushed the MCU send the defa possible terminal lockup. There are two reset n CPU32Bug default, refer to the **RESET** com system initialization occurs, similar to the BC restored to their default states. The serial port is cleared. The offset registers are cleared. The ta character queues are cleared. On-board devices (reset, CPU32Bug variables and tables are pres breakpoints.

Use reset if the processor halts, for example, a environment is lost (vector table is destroyed, etc

1.5.2 Abort

The M68300PFB platform board abort switch te is executed while running target code, a snapsho the target registers. For this reason abort is appr being debugged. Use abort to regain control if th PC, stack pointers, etc. help pinpoint malfunction

Abort generates a non-maskable, level-seven in state at the time of an abort and are displayed on the user code are removed and the breakpoint tab debugger.

CHAP

GENERAL IN

1.1 INTRODUCTION

This chapter provides a general description, inst instructions, memory requirements, and a termin M68CPU32BUG Debug Monitor (hereafter re manual covers the 1.00 version of the CPU32Bu

1.2 GENERAL DESCRIPTION

The CPU32Bug package evaluates and debugs s Card Computer. System evaluation facilities programs. Various CPU32Bug routines that hand available to user programs through the TRAP #1

CPU32Bug includes:

- Commands for display and modificati
- Breakpoint capabilities,
- An assembler/disassembler useful for
- A power-up self test feature which ve
- A command-driven user-interactive so
- A user interface which accepts comm

There are two modes of operation in the CPU diagnostic mode. When the user is in the d displayed, and the user has access to the debugg in the diagnostic mode the prompt CPU32Diag diagnostic commands (see Chapter 6). These mo

CPU32Bug is command-driven. It performs var entered at the keyboard. Figure 1-1 illustrates executes entered commands and the prompt reap is entered which causes execution of user targe return to CPU32Bug. This depends upon the user

CPU32Bug is similar to Motorola's other debu differences. Many of the commands are more debugger has more detailed error messages and a

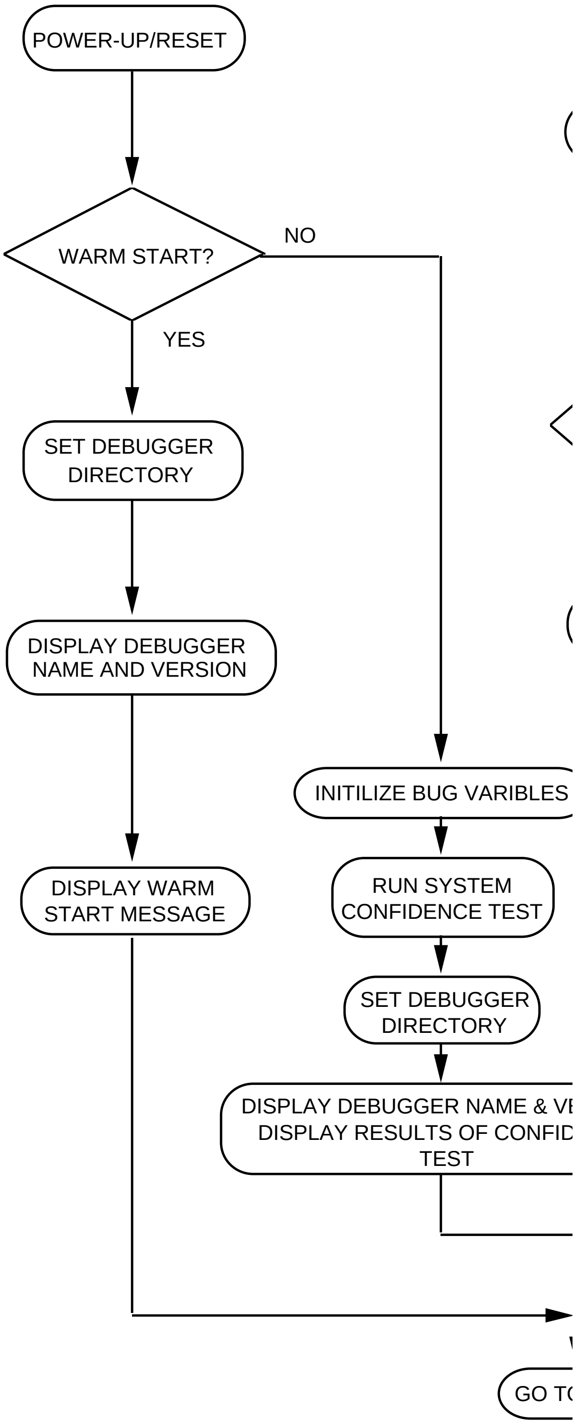


Figure 1-1. CPU32Bug Ope

1.3 USING THIS MANUAL

Those users unfamiliar with debugging packages
CPU32Bug. This provides information about CP

Paragraph 1.4 Installation and Start-up describe
module and obtaining the CPU32Bug prompt on

For questions about syntax or operation of a
paragraph which describes that particular comma

Some debugger commands take advantage of th
command descriptions in Chapter 3 assu
assembler/disassembler functionality. Chapter
disassembler.

NO

In the examples shown, all user inputs a
the examples by distinguishing between
output by CPU32Bug. The symbol <**CR**>
user's terminal keyboard. Whenever this
return should be entered by the user.

1.4 INSTALLATION AND START-U

Use the following set-up procedure to enable CP

1. Configure the jumpers on the BC
   Motorola publication number M6833
2. Connect the DB-9 serial communic
   computer which is to be the CPU32B
   cable to P4 on the BCC.

   Set up the terminal as follows:

   - Eight bits per character
   - One stop bit per character
   - Parity disable
   - 9600 baud rate

1.5.3 Break

The **BREAK** key on the terminal keyboard
interrupt. The only time break is recognized i
debugger console. Break removes any breakpoint
intact. Break does not, however, take a snapsh
target registers. It is useful for terminating activ
blocks of data.

NO

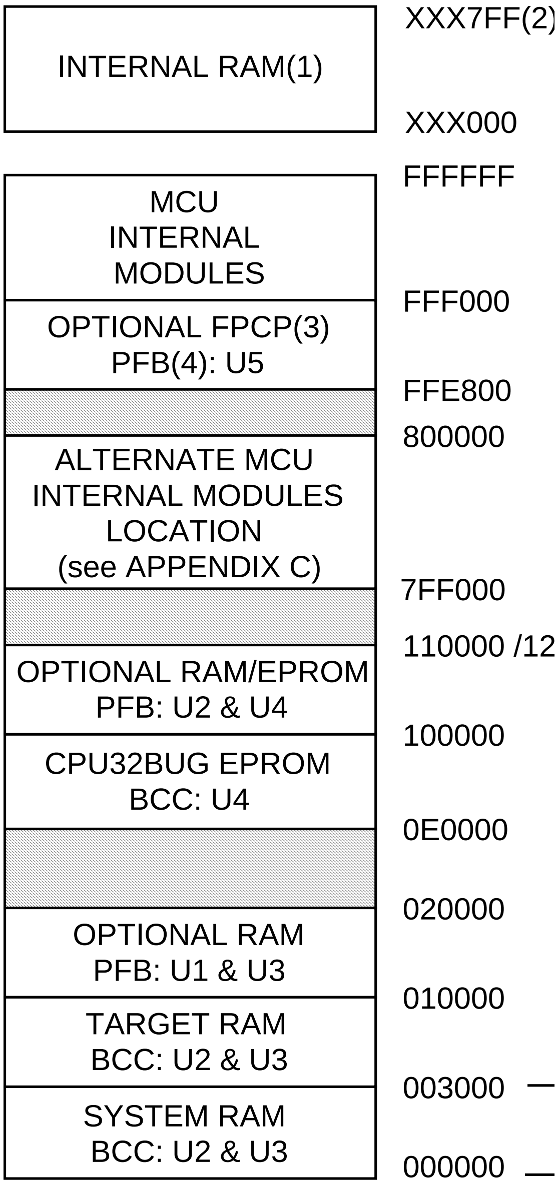
When using terminal emulation
Kermit, the BREAK key on the k
program and may not be transm
emulation program's user manual
BREAK signal to the port connec

1.6 MEMORY REQUIREMENTS

The program portion of CPU32Bug is approxim
the BCC contains 128k bytes and is mapped
CPU32Bug code is position-independent and can
of the EPROM ($F0000 - $FFFFF) is blank an
CPU32Bug Customization.

CPU32Bug requires a minimum of 12k bytes of
memory may be either off-board system memory
board RAM. On-board RAM allows stand-alone

The first 12k bytes are used for CPU32Bug s
memory is reserved as user space. Whenever t
initialized to the beginning user space address
addresses at the end of the user space. The targe
of the user space. Register initialization is done s
CPU32 Reference Manual for information reg
on/reset.



(1) Consult the MCU device User's Manu
(2) XXXBase address is user programm
such as internal RAM, can be config
using the Initialization Table (INITTB
(3) Floating Point Coprocessor - MC688
(4) Platform Board
(5) Depends on the memory device type

Figure 1-2. BC

1.7 TERMINAL INPUT/OUTPUT CO

When entering a command at the prompt, the f
preceding the character, this indicates that the C
striking the character key).

^X (Cancel line)	The cursor is backspac
^H (backspace)	The cursor is moved cursor position is erase
<del> (delete/rubout)	Performs the same fun
^D (redisplay)	The entire command line.

When observing output from any CPU32Bug co
entered to control the output, if the XON/XOFF
are initialized to "^S" and "^Q" respectively by C
the **PF** command. The initialized (default) mode

^S (wait)	Console output is halte
^Q (resume)	Console output is resu

EXAMPLES Valid expressions.

Expression
FF0011
45+99
&45+&99
@35+@67+@10
%10011110+%1001
88<<10
AA&F0

The total value of the expression must be betwee

2.2.1.2 Address as a Parameter

Many commands use <ADDR> as a parameter. the one accepted by the MC68300 Family one-l allowed. An address+offset register mode is also

Table 2-1 summarizes the address formats w debugger command lines.

CHAP

DEBUG MONITO

2.1 INTRODUCTION

CPU32Bug performs various operations in respo When the debugger prompt CPU32Bug> appears accept commands.

2.2 ENTERING DEBUGGER COMM

As the command line is entered it is stored in ar carriage return is entered. This allows the user to described in paragraph 1.7.

The debugger executes commands and returns th command causes execution of user target code, (the debugger. This depends upon the user prog specified, then control returns to the debugger program also returns control to the debugger b (described in paragraph 5.2.16). Also refer to the of the **GO** commands.

In general debugger commands include:

- A command identifier (i.e., **MD** or upper- or lower-case characters are al
- At least one intervening space before
- A port number for running with multi
- Any required arguments, as specified
- An option field, set off by a semicolo conditions of the command.
- Some commands (**MD**, **GO**, **T**, etc) (<**CR**>) only causes the last comman any, incremented. Thus after an **MD** displayed by entering a carriage r command, entering a carriage return (
- Multiple debugger commands may be the commands with the explanation p

The commands use a modified Backus-Naur syn

<> The angular brackets enclose syntactic variable is repl symbols it represents.

[] Square brackets enclose occur zero or one time. I required characters.

[]... Square brackets follow optional/repetitive. The s more times.

| This symbol indicates tha symbols separated by a str

/ Select one or more of the s

{} Brackets enclose optional

2.2.1 Syntactic Variables

The following syntactic variables are used in addition, other syntactic variables may be use description in which they occur.

<DEL> Delimiter; either a comma detail in paragraph 2.2.1.1

<ADDR> Address (described in deta

<COUNT> Count; the same syntax as

<RANGE> A range of memory add ADDR><DEL><ADDR>

<TEXT> An ASCII string of as man marks ('TEXT').

2.2.1.1 Expression as a Parameter

An expression is one or more numeric values sep

- \+ plus
- – minus
- * multiplied by
- / divided by
- & logical AND
- << shift left
- >> shift right

Base identifiers define numeric values as either a

Base	Identif
Hexadecimal	$
Decimal	&
Octal	@
Binary	%

If no base identifier is specified, then the numeri

A numeric value may also be expressed as a s string literal must begin and end with single quo the concatenation of the ASCII values of the ch other numeric value.

String Literal	
'A'	
'ABC'	
'TEST'	

Evaluation of an expression is always from left t of the expression. There is no operator prece evaluated first. Nested parenthetical sub-expressi

2.5.1 CPU32Bug Vector Table and Workspac

CPU32Bug requires 12k bytes of RAM to oper this memory space. The first 1024-bytes are rese second 1024-bytes are reserved as an exceptic CPU32Bug reserves space for static variables an values. After the static variables, CPU32Bug all the system stack pointer to the top of this area.

With the exception of the first 1024-byte vecto reserved memory areas. Refer to paragraph 1.6 for example, a user program inadvertently wrote communication parameters, these parameters wo with the system terminal. If a user program co may be loaded into the processor's counter, caus

2.5.2 CPU32Bug Exception Vectors

The debugger exception vectors are listed below the target program vector table or the associat etc.) will not operate.

Table 2-2. CPU32Bu

Vector Number	Offset	Exception	
4	$10	Illegal	Ir G
9	$24	Trace	T
31	$7C	Level 7 interrupt	A
47	$BC	TRAP #15	S
66	$108	User Defined	T

When the debugger handles one of the exceptio left pointing past the bottom of the exception pointer values just before the exception occur facility (through an exception) is transparent to stack.

Table 2-1. Debugger Ad

Format	Example	
N	140	Absolute addres
N+Rn	332+R5	Absolute addres assembler-acce
(An)	(A1)	Address register
(d,An) or d(An)	(120,A1) 120(A1)	Address register
(d,An,Xn) or d(An,Xn)	(&120,A1,D2) &120(A1,D2)	Address register accepted).
Symbol D		
	N - Absolute address (any valid expr Dn - Data register n An - Address register n Xn - Index register n (An or Dn) d Dis bd - Base displacement (any valid ex Rn - Offset register n ZXn - Zero suppressed register Xn	

2.2.1.3 Offset Registers

Eight pseudo-registers (R0 through R7) called o of re-locatable and position-independent files. (normally 0), but when loaded into memory, d different memory location. Implementing offset the listing with addresses in the loaded program. into account this difference and forcing the d format. The range for each offset register is set base and top addresses for an offset register sets which may overlap. In the event that an addres one that yields the least offset is chosen.

NO

Relative addresses are limited to of the range of the closest offset r

EXAMPLE A portion of the listing fi
MC68300 Family DOS re:

```
1
2
3
4
5       0  00000000  48E78080
6       0  00000004  4280
7       0  00000006  1018
8       0  00000008  5340
9       0  0000000A  12D8
10      0  0000000C  51 C8FFFC
11      0  00000010  4CDF0101
12      0  00000014
13
14
******  TOTAL ERRORS 0-
******  TOTAL WARNINGS
```

The above program was loaded at address 00004

```
CPU32Bug>MD 427C;DI<CR>
0000427C  48E78080       MOVEM.L
00004280  4280           CLR.L
00004282  1018           MOVE.B
00004284  5340           SUBQ.W
00004286  12D8           MOVE.B
00004288  51C8FFFC       DBF
0000428C  4CDF0101       MOVEM.L
00004290  4E75           RTS
```

By using one of the offset registers, the disasse
listing file address as follows:

```
CPU32Bug>OF R0<CR>
R0 =00000000 00000000? 427C: 16.<
CPU32Bug>MD 0+R0;DI<CR>
00000+R0  48E78080       MOVEM.L
00004+R0  4280           CLR.L
00006+R0  1018           MOVE.B
00008+R0  5340           SUBQ.W
0000A+R0  12D8           MOVE.B
0000C+R0  51C8FFFC       DBF
00010+R0  4CDF0101       MOVEM.L
00014+R0  4E75           RTS
CPU32Bug>
```

For Additional information about the offset regis

2.2.2 Port Numbers

Some CPU32Bug commands allow the user to
Valid port numbers are:

0 - MCU SCI Port (RS-232C communica

Although CPU32Bug supports other ports (see
the BCC to support additional ports. Thus the c
PF, **VE**) can only use port 0. Those comman
functional without additional hardware.

2.3 ENTERING AND DEBUGGING P

There are various ways to enter a user program in
using the assembler/disassembler option and the

The user enters the program one source line at
assembled and the object code is loaded into me
the CPU32Bug assembler/disassembler.

Another way to enter a program is to download a
computer). The program must be in S-record f
assembled or compiled on the host system. Th
memory via the debugger **LO** command. Alte
CPU32Bug **MM** command as outlined above
command. A communication link must exist betv

2.4 CALLING SYSTEM UTILITIES F

A convenient method to input and output chara
provided by the TRAP #15 instructions. This fr
into the target code. Refer to Chapter 5 for det
execute them from a user program.

2.5 PRESERVING DEBUGGER OPER

Avoiding contamination of the debugger opera
paragraphs. CPU32Bug uses certain MCU on-bo
memory to store temporary variables, exceptio
dependent memory space, then the debugger may

Before the normal register display informatior displayed. This includes the type of exception wi

Mnemonic	Des
SSW	Special S
Fault Addr.	Faulted A
Data	Data
Cur. PC	Program
Cnt. Reg.	Internal T Count Re

The upper nibble of the count register (Cnt. Reg. MCU device. Consult the CPU32 Reference Ma details.

Notice that the target stack pointer is different. value of the stacked exception stack frame. Ex command.

```
CPU32Bug>MD (A7):C<CR>
00003FE8  A700 0000 3000 C008   0(
00003FF8  0000 3000 0001 0065
CPU32Bug>
```

2.6 FUNCTION CODE SUPPORT

Function codes identify the address space bein extension of the address. The function codes pro proper memory location.

For this reason, all debugger commands involv specification of function codes:

The caret (^) symbol following the address fi follows. The function code can be entered by sp specifying a number between 0 and 7. The synt are:

<ADDR>^<FC>	Sets the function code to <
<ADDR>^^	Toggles the displaying of
<ADDR>^<FC>=	Sets the function code to The default value at power

EXAMPLE Trace one instruction using

```
CPU32Bug>RD<CR>
PC   =00003000     SR   =2700=TR:OF
SFC  =5=SD         DFC  =5=SD
D0   =00000000     D1   =00000000
D4   =00000000     D5   =00000000
A0   =00000000     A1   =00000000
A4   =00000000     A5   =00000000
00003000 203900100000          MOV

CPU32Bug>T<CR>
PC   =00003006     SR   =2700=TR:OF
SFC  =5=SD         DFC  =5=SD
D0   =12345678     D1   =00000000
D4   =00000000     D5   =00000000
A0   =00000000     A1   =00000000
A4   =00000000     A5   =00000000
00003006 D280        ADD.L       D0,
CPU32Bug>
```

Notice that the value of the target stack pointer r exception has taken place. The user program m CPU32Bug or it may create a separate exception

2.5.2.1 Using CPU32Bug Target Vector Table

CPU32Bug initializes and maintains a vector tab any user program started by the CPU32Bug with this target-vector table area is the base address address is loaded into the target-state-vector base For verification use the **RD** command immedi registers.

CPU32Bug loads the target-vector table with th other vector locations with the address of a ge 2.5.2.3). The target program allocates as many exception vectors into the table. If the vector loc the accompanying debugger functions will be los

CPU32Bug maintains a separate vector table fo memory space. The debugger vector table is modifications should ever be made to it.

2.5.2.2 Creating Vector Tables

A user program may create a separate vector tal the user program must change the value of the table. To use the debugger facilities, copy the v corresponding user vector table locations (block

The vector for the CPU32Bug generalized exce 2.5.2.3) may be copied from offset $08 (Bus locations in the user's vector table where a separ diagnostic support in the event execution of th exception. The generalized exception handler giv identifies the type of the exception.

The following is an example of a user routine w the vector base register to point at it.

```
*
***     BUILDX – Build exception v
*
BUILDX    MOVEC.L     VBR,A0
          LEA         $1 0000,A1
          MOVE.L      $8(A0),D0
          MOVE.W      $3FC,D1
LOOP      MOVE.L      D0,(A1,D1)
          SU BQ.W     #4, D 1
          BPL.B       LOOP
          MOVE.L      $1 0(A0),$1
          MOVE.L      $24(A0),$24
          MOVE.L      $BC(A0),$BC
          LEA.L       TIMER(PC),A
          MOVE.L      A2,$2C(A1 )
          MOVEC.L     A1 ,VBR
          RTS
          END
```

The user program may use one or more of the e operation if the user's exception handler can de when to pass the exception to the debugger.

When an exception occurs which requires c exception handler must read the vector offset fro offset is added to the address of the CPU32Bu program saves), producing the address of the C then jumps to the address stored at this vect CPU32Bug exception handler).

The user program must ensure an exception stac one the processor would create for the particula the exception handler.

EXAMPLE The user exception handle

```
*
*** EXCEPT - Exception handler **
*

EXCEPT    SUBQ.L    #4,A7
          LINK      A6,#0
          MOVEM.L   A0-A5/D0-D7,-(A

          : decide here if user code will handl

          MOVE.L    BUFVBR,A0
          MOVE.W    14(A6), Do
          AND.W     #$0FFF,D0
          MOVE.L    (A0,D0.W),4(A6)
          UNLK
          RTS
```

2.5.2.3 CPU32Bug Generalized Exception Ha

The CPU32Bug generalized exception handler s For these exceptions, the target stack pointer poi In this way, if an unexpected exception occurs du stack frame displays to assist in determining the

EXAMPLE Bus error at address $F000 of memory location $F000

```
CPU32Bug>RD<CR>
PC   =00003000    SR   =2700=TR:OFF_S_7_
SFC  =5=SD        DFC  =5=SD        USP
D0   =00000000    D1   =00000000    D2
D4   =00000000    D5   =00000000    D6
A0   =00000000    A1   =00000000    A2
A4   =00000000    A5   =00000000    A6
00003000 203900F0    0000       MOVE.L
CPU32Bug>T<CR>
Exception: Bus Error
Format/Vector=C008
SSW=0065  Fault Addr.=00F00000  Data=FFF
PC   =00003000    SR   =A700=TR:ALL_S_7_
SFC  =5=SD        DFC  =5=SD        USP
D0   =00000000    D1   =00000000    D2
D4   =00000000    D5   =00000000    D6
A0   =00000000    A1   =00000000    A2
A4   =00000000    A5   =00000000    A6
00003000 203900F0    0000       MOVE.L
CPU32Bug>
```

Table 3-1. Debug Monito

Command Mnemonic	Ti
OF	Offset Registers Display/M
PA/NOPA	Printer Attach/Detach
PF	Port Format
RD	Register Display
RESET	Cold/Warm Reset
RM	Register Modify
RS	Register Set
SD	Switch Directories
T	Trace
TC	Trace On Change of Contr
TM	Transparent Mode
TT	Trace To Temporary Brea
VE	Verify S-Records Against

Each command is described in the following te section 2.1. In the examples of the debugger c helps clarify examples by distinguishing user inp The symbol <**CR**> represents the carriage ret symbol indicates the user should enter a carriage

The valid function code mnemonics are:

Function	Code Mnemon
0	F0
1	UD
2	UP
3	F3
4	F4
5	SD
6	SP
7	CS

The **BR**, **GD**, **GO**, and **GT** commands set the va or supervisor program (SP). When execution is space is determined by bit 13 (the S-bit) of the s cleared, UP is used. By specifying a function coc is forced to the correct state before execution beg

For the **GT** command, the temporary breakpoir defaults to SP or UP, depending on the state of th

Though function codes are supported, the BCC operate.

EXAMPLE To change data at location

```
CPU32Bug>m 5000^ud<CR>
00005000^UD   0000   ?   1234.<CR>
CPU32Bug>
```

CHAP

DEBUG MONITO

3.1 INTRODUCTION

This chapter contains descriptions and examples
3-1 summarizes these commands.

Table 3-1. Debug M

Command Mnemonic	Tit
BC	Block of Memory Compare
BF	Block of Memory Fill
BM	Block of Memory Move
BR/NOBR	Breakpoint Insert/Delete
BS	Block of Memory Search
BV	Block of Memory Verify
DC	Data Conversion
DU	Dump S-Records
GD	Go Direct (Ignore Breakpo
GN	Go to Next Instruction
GO	Go Execute User Program
GT	Go To Temporary Breakpo
HE	Help
LO	Load S-Records from Hos
MA/NOMA	Macro Define/Display/Dele
MAE	Macro Edit
MAL/NOMAL	Macro Expansion Listing E
MD	Memory Display
MM	Memory Modify (alias M)
MS	Memory Set

```
CPU32Bug>BF 4000:10 4E71 ;B<CR>
Effective address: 00004000
Effective count  : &16
Truncated data = $71
CPU32Bug>MD 4000:30;B<CR>
00004000  71 71 71 71 71 71 71 71  71 7
00004010  00 00 00 00 00 00 00 00  00 0
00004020  00 00 00 00 00 00 00 00  00 0
CPU32Bug>
```

The specified data did not fit into the specified ''Data = '' message was output.

```
CPU32Bug>BF 4000,4006 12345678 ;L<CR>
Effective address: 00004000
Effective address: 00004003
CPU32Bug>MD 4000:30;B<CR>
00004000  12 34 56 78 00 00 00 00  00 0
00004010  00 00 00 00 00 00 00 00  00 0
00004020  00 00 00 00 00 00 00 00  00 0
CPU32Bug>
```

The longword pattern would not fit evenly in tl and the "Effective address" messages reflect the the specified address.

```
CPU32Bug>BF 4000:18 0 1<CR>
Effective address: 00004000
Effective count  : &24
CPU32Bug>MD 4000:18<CR>
00004000 0000 0001 0002 0003  0004 0005
00004010 0008 0009 000A 000B  000C 000D
00004020 0010 0011 0012 0013  0014 0015
```


3.2 BLOCK OF MEMORY COMPAR

BC <range><del><addr>[;B|W|L]

options:

B – Byte

W – Word

L – Longword

The **BC** command compares the contents of the r place in memory, beginning at <addr>.

The option field is only allowed when <range> i or L defines the size of data to which the count i option of L would mean to compare four long w range beginning address is greater than the end an option field is specified without a count in the

For the following examples, assume the followin

```
CPU32Bug>MD 4000:20;B<CR>
00004000   54 48 49 53 20 49 53 20   41
00004010   00 00 00 00 00 00 00 00   00

CPU32Bug>MD 4100:20;B<CR>
00004100   54 48 49 53 20 49 53 20   41
00004110   00 00 00 00 00 00 00 00   00
```

EXAMPLES

```
CPU32Bug>BC 4000,401F 4100<CR>
Effective address: 00004000
Effective address: 0000401F
Effective address: 00004100
CPU32Bug>                          Mem
```

```
CPU32Bug>BC 4000:20 4100;B<CR>
Effective address: 00004000
Effective count  : &32
Effective address: 00004100
CPU32Bug>                          Mem
```

```
CPU32Bug>MM 410F;B<CR>
0000410F  21? 0.<CR>
CPU32Bug>                          Cre
```

```
CPU32Bug>BC 4000:20 4100;B<CR>
Effective address: 00004000
Effective count  : &32
Effective address: 00004100
0000400F: 21   0000410F: 00
CPU32Bug>                          Mis
```


3.3 BLOCK OF MEMORY FILL

BF <range><del><data>[<del><incremen

where:

<data> and <increment> are both express

options:

B – Byte

W – Word

L – Longword

The **BF** command fills the specified range of r specified, then <data> is incremented by this remains a constant value. Enter a negative value pattern . The data entered by the user is right-jus specified by the option selected. The default field

User-entered data or increment must fit into the truncation occurs, then a message is printed s increment value.

If the range is specified using a count then the co

Truncation always occurs on byte or word size example, entering "-1" internally becomes $FFF $FFFF for word sized fields. There is no differen $FFFFFFFFF, so truncation occurs for byte or w

If the upper address of the range is not on the cor to be written, then data is filled to the last bound of the specified range are not written under displayed by the command show the extent of the

EXAMPLES Assume memory from $4000 t

```
CPU32Bug>BF 4000,401F 4E71<CR>
Effective address: 00004000
Effective address: 0000401F
CPU32Bug>MD 4000 402F<CR>
00004000  4E71 4E71 4E71 4E71  4E71 4E7
00004010  4E71 4E71 4E71 4E71  4E71 4E7
00004020  0000 0000 0000 0000  0000 000
```

Since no option was specified, the length of the d

3.6 BLOCK OF MEMORY SEARCH

BS <range><del><text> [;B|W|L] or

BS <range><del><data>[<del><mask>]

The **BS** command searches the specified range (
pattern. This command has three modes:

Mode 1 LITERAL STRING SEARCH —
the literal string entered by the us
by a <text> field. The size as sp
count field in <range> refers to
available only if <range> is speci
address of the first byte of the ma

Mode 2 DATA SEARCH — a data patter
line. The data field size is entere
word (W). The size entered in the
in <RANGE> refers to bytes, wor
data search:

1. The user-entered data patt
or leading zeros are add
specified size.
2. Successive bytes, words,
compared to the user-ente
at bit positions correspond
then a default mask of all
the mask is the same size a
3. If the "N" (non-aligned) op
on a byte-by-byte basis, r
the size of <data>. This is
sought, but is not expected
4. If a match is found, the a
along with the memory co
at the memory location is

Mode 3 DATA VERIFICATION — If the
contents do not match the user-
displayed. Otherwise this mode is

3.4 BLOCK OF MEMORY MOVE

BM <range><del><addr> [;B|W|L]

options:

B – Byte

W – Word

L – Longword

The **BM** command copies the contents of the me
place in memory, beginning at <addr>. The c
specified using a count. In this case the B, W, or
referring. For example, a count of four with an o
(or 16 bytes) to the new location. An error results
the range.

EXAMPLES Assume memory from $4000

```
CPU32Bug>MD 4100:20;B<CR>
00004100  544B 4953 2049 5320  4120 544
00004110  0000 0000 0000 0000  0000 000

CPU32Bug>BM 4100 410F 4000<CR>
Effective address: 00004100
Effective address: 0000410F
Effective address: 00004000

CPU32Bug>MD 4000:20;B<CR>
00004000  5448 4953 2049 5320  4120 544
00004010  0000 0000 0000 0000  0000 000
```

This utility is useful for patching assembly co
program in memory at address $6000.

```
CPU32Bug>MD 6000 600A;DI<CR>
00004000 D480                      ADD.L
00004002 E2A2                      ASR.L
00004004 2602                      MOVE.L
00004006 4E4F0021                  SYSCALL
0000400A 4E71                      NOP
```

Now suppose the user would like to insert an
ASR.L instruction. Block move the object code c

```
CPU32Bug>BM 6002 600B 6004<CR>
Effective address: 00006002
Effective address: 0000600B
Effective address: 00006004
CPU32Bug>MD 6000 600C;DI<CR>
00006000 D480                       ADD
00006002 E2A2                       ASR
00006004 E2A2                       ASR
00006006 2602                       MOV
00006008 4E4F                       SYS
0000600C 4E71                       NOP
```

Now the user need simply enter the NOP at addr

```
CPU32Bug>MM 6002;DI <CR>
00006002 E2A2                ASR.L
00006002 4E71                NOP
00006004 E2A2                ASR.L
CPU32Bug>
```

```
CPU32Bug>MD 6000 600C;DI<CR>
00006000 D480                ADD.L
00006002 4E71                NOP
00006004 E2A2                ASR.L
00006006 2602                MOVE.L
00006008 4E4F                TRAP
0000600C 4E71                NOP
CPU32Bug>
```


3.5 BREAKPOINT INSERT/DELETE

BR {<addr>[:<count>]}

NOBR [<addr>]

The **BR** command allows the user to set a target
for debugging purposes. Enter only the **BR** com
breakpoint table, or enter {<addr> [:<count>]}
during target code execution a breakpoint with 0
the target registers and control returned to CPU3
of the processor at selected instructions in the co

Breakpoints are normally only used in RAM, b
under the TRACE commands (see **T**, **TC**, and **T**

As many as eight breakpoints can be defined. Br
each time either BR or NOBR is used. If an ad
address is added to the breakpoint table. The cou
at the breakpoint address is fetched before a b
hexidecimal input, unless a numeric identifier p
decremented with each fetch. Every time a bre
handler routine prints the CPU state on the scr
maximum <count> is a 32-bit value ($FFFFFFFF

NOBR is used to delete breakpoints from the
from the breakpoint table, enter **NOBR** followed
all entries are deleted from the breakpoint table a

EXAMPLE

```
CPU32Bug>BR 4000,4200 4700:&12 <CR>
BREAKPOINTS
00004000            00004200
00004700:C

CPU32Bug>NOBR 4200 <CR>
BREAKPOINTS
00004000            00004700:C

CPU32Bug>NOBR <CR>
BREAKPOINTS
CPU32Bug>
```

EXAMPLES Assume memory from $6000

```
CPU32Bug>MD 6000:30;B <CR>
00006000 4E71 4E71 4E71 4E71  4E71 4E71
00006010 4E71 4E71 4E71 4E71  4E71 4E71
00006020 4E71 4E71 4E71 4E71  4E71 4E71
CPU32Bug>BV 6000 601F 4E71 <CR>      Defa
Effective address: 00006000
Effective address: 0000601F
CPU32Bug>
```

Assume memory from $5000 to $502F is as indi

```
CPU32Bug>MD 5000:30;B<CR>
00005000 0000 0000 0000 0000  0000 0000
00005010 0000 0000 0000 0000  0000 0000
00005020 0000 0000 0000 0000  0000 4AFB
CPU32Bug>BV 5000:30,0;B<CR>
Effective address: 00005000
Effective count  : &48
0000502A|4A   0000502B|FB    0000502C|4
0000502E|4A   0000502F|FB
CPU32Bug>
```

Assume memory from $7000 to $702F is as indi

```
CPU32Bug>MD 7000:18 <CR>
00007000 0000 0001 0002 0003  0004 0005
00007010 0008 FFFF 000A 000B  000C 000D
00007020 0010 0011 0012 0013  0014 0015
CPU32Bug>BV 7000:18,0,1 <CR>
Effective address: 00007000
Effective count  : &24
00007012|FFFF
CPU32Bug>
```


In all three modes information on matches is out
24 lines of matches are displayed on the screen
screen indicating there are more lines to display.
the **BREAK** key to cancel the output and exit the

If a match (or a mismatch in the case of Mode
whose beginning is within and end is outside o
output stating that the last match does not lie ent
contiguous memory with this command without

EXAMPLES Assume the following data is

```
00003000 0000 0045 7272 6F72  2053 7461
00003010 3446 2F2F 436F 6E66  6967 5461
00003020 7461 7274 3A00 0000  0000 0000
```

```
CPU32Bug>BS 3000 302F 'Task Status'<CR>
Effective address: 00003000
Effective address: 0000302F
-not found-
```

```
CPU32Bug>BS 3000 302F 'Error Status'<CR
Effective address: 00003000
Effective address: 0000302F
00003003
```

```
CPU32Bug>BS 3000 301F 'ConfigTableStart
Effective address: 00003000
Effective address: 0000301F
00003014
-last match extends over range boundary
```

```
CPU32Bug>BS 3000:30 't' ;B<CR>
Effective address: 00003000
Effective count  : &48
0000300A  0000300C  00003020  00003023
```

```
CPU32Bug>BS 3000:18,2F2F<CR>
Effective address: 00003000
Effective count  : &24
00003012|2F2F
```

```
CPU32Bug>bs 3000,302F 3d34<CR>
Effective address: 00003000
Effective address: 0000302F
-not found-
```

```
CPU32Bug>bs 3000,302F 3d34 ;n<CR>
Effective address: 00003000
Effective address: 0000302F
0000300F|3D34
```

```
CPU32Bug>BS 3000:30 60,F0 ;B<CR>
Effective address: 00003000
Effective count  : &48
00003006|6F   0000300B|61   00003015|6F
00003017|66   00003018|69   00003019|67
0000301C|62   0000301D|6C   0000301E|65
```

```
CPU32Bug>BS 3000 302F 0 F;V<CR>
Effective address: 00003000
Effective address: 0000302F
00003002|0045   00003004|7272   0000300
0000300A|7461   0000300C|7475   0000300
00003012|2F2F   00003014|436F   0000301
0000301A|5461   0000301C|626C   0000301
00003022|7274
```


3.7 BLOCK OF MEMORY VERIFY

BV <range><del><data> [<del><increme

where:

<data> and <increment> are both express

options:

B – Byte
W – Word
L – Longword

The **BV** command compares the specified ra
increment is specified, then <data> is increme
otherwise <data> remains a constant value. Enter
pattern. The data entered by the user is right-ju
length (as specified by the option selected). The

User-entered data or increment must fit into the
truncation occurs, then a message is printed s
increment value.

If the range is specified using a count then the co

Truncation always occurs on byte or word size
example, entering "-1" internally becomes $FFF
$FFFF for word sized fields. There is no differen
$FFFFFFFFF, so truncation occurs for byte or w

If the upper address of the range is not on the co
to be verified, then data is verified to the last
outside of the specified range are not read und
displayed by the command show the extent of the

Enter ALT-F1 again to close the log file TES' "Effective address" and "CPU32Bug", but they v commands, as it keys on the "S" character. The so desired.

3.8 DATA CONVERSION

DC <exp>I<addr>

Use the **DC** command to simplify an expressio value is displayed in its hexadecimal and de interpreted as a signed negative number (i.e., i representation of the number is set) then bot displayed.

Use **DC** to obtain the equivalent effective addres

EXAMPLES

```
CPU32Bug>DC 10<CR>
        00000010 = $10 = &16

CPU32Bug>DC &10-&20<CR>
SIGNED : FFFFFFF6 = -$A = -&10
UNSIGNED: FFFFFFF6 = $FFFFFFF6 = &42949

CPU32Bug>DC 123+&345+@67+%1100001<CR>
        00000314 = $314 = &788

CPU32Bug>DC (2*3*8)/4<CR>
        0000000C = $C = &12

CPU32Bug>DC 55&F<CR>
        00000005 = $5 = &5

CPU32Bug>DC 55>>1<CR>
        0000002A = $2A = &42
```

The subsequent examples assume A0=00003000

```
00003000  11111111   22222222   333333

CPU32Bug>DC (A0)<CR>
        00003000 = $3000 = &12288

CPU32Bug>DC ([A0])<CR>
        11111111 = $11111111 = &2863311

CPU32Bug>DC (4,A0)<CR>
        00003004 = $3004 = &12292

CPU32Bug>DC ([4,A0])<CR>
        22222222 = $22222222 = &5726623
```

3.9 DUMP S-RECORDS

DU [<port><del>]<range><del>[<text><

The **DU** command outputs data from memory
specified by the user. If <port> is not specified th
For S-record information see Appendix A.

The option field is only allowed when <range> i
or L defines the size of data to which the count i
option of L would mean to move four longwor
results if an option field is specified without a co

Use the optional <text> field for incorporating
records that is to be dumped.

To use the optional <addr> field, enter an ent
records. This address is incorporated into the add
entry address is entered then the address fiel
beginning <range> address. The termination reco
the entered address.

An optional offset may also be specified by the
added to the addresses of the memory locations b
is written to the address field of the S-records, cr
memory at a different location than that from wh

NO

If an offset is specified but no en
commas (indicating a missing f
keep it from being interpreted as a

EXAMPLES Dump memory from $8000 to

```
CPU32Bug>DU 1 8000 802F<CR>
Effective address: 00008000
Effective address: 0000802F
CPU32Bug>
```


Dump 10 bytes of memory beginning at $3000 to

```
CPU32Bug>DU 3000:&10;B<CR>
Effective address: 00003000
Effective count  : &10
S0003000FC
S10D3000000000040008000C00109A
S9030000FC
CPU32Bug>
```

Dump memory from $4000 to $402F to host (p
header record and specify an entry point of $400

```
CPU32Bug>DU 1 4000 402F 'TEST' 400A<CR>
Effective address: 00004000
Effective address: 0000402F
CPU32Bug>
```

The following example illustrates the procedure
this case an IBM-PC or compatible running M
utility. Assume memory from $4000 to $4007 is

```
CPU32Bug>MD 4000:4;DI<CR>
00004000  7001                    MOV
00004002  D089                    ADD
00004004  4A00                    TST
00004006  4E75                    RTS
CPU32Bug>
```

Enter the following command to dump S-record
start address of $4000, a title of 'TEST.MX', an
<CR> to send the **DU** command to CPU32Bug,
to open a log file. Enter the filename as TEST
complete the **DU** command entry. The DU comm
copies it into the file TEST.MX.

```
CPU32Bug>DU 4000 4007 'TEST.MX' 4000 65
Effective address: 00004000
Effective address: 00004007
S00A0000544553542E4D58E2
S30D650040007001D089A004E7576
S7056500400055
CPU32Bug>
```

Use the **GN** command to trace through the subro

```
CPU32Bug>GN<CR>
Effective address: 00006008
Effective address: 00006004
At Breakpoint
PC    =00006008     SR    =2700=TR:OFF_S_
SFC   =0=F0         DFC   =0=F0         US
D0    =00000004     D1    =00000001     D2
D4    =00000000     D5    =00000000     D6
A0    =00000000     A1    =00000000     A2
A4    =00000000     A5    =00000000     A6
00006008   2600                       MOVE.
CPU32Bug>
```


3.10 GO DIRECT (IGNORE BREAK

GD [<addr>]

Use the **GD** command to start target code execu
target PC. Execution starts at the target PC addr
inserted.

Once execution of target code begins, control is

- Press the ABORT switch or RESET s
- Execute the .RETURN TRAP #15 fu
- Generation of an unexpected exceptio

EXAMPLE The following program reside

```
CPU32Bug>MD 4000;DI<CR>
00004000    2200          MOVE.L     D0,
00004002    4282          CLR.L      D2
00004004    D401          ADD.B      D1,
00004006    E289          LSR.L      #$1
00004008    66FA          BNE.B      $40
0000400A    E20A          LSR.B      #$1
0000400C    55C2          SCS.B      D2
0000400E    60FE          BRA.B      $40
CPU32Bug>RM D0<CR>
```

Initialize D0 and start target program:

```
D0    =00000000 ? 52A9C.<CR>
CPU32Bug>GD 4000<CR>
Effective address: 00004000
```

To exit target code, press **ABORT** pushbutton.

```
Exception: Abort
PC   =0000400E      SR   =2711=TR:OFF_S_
SFC  =0=F0          DFC  =0=F0
D0   =00052A9C      D1   =00000000
D4   =00000000      D5   =00000000
A0   =00005000      A1   =00000000
A4   =00000000      A5   =00000400
0000400E    60FE                    BRA.B
CPU32Bug>
```

Set PC to start of program and restart target code

```
CPU32Bug>RM PC<CR>
PC    =0000400E ? 4000.<CR>
CPU32Bug>GD<CR>
Effective address: 00004000
```


3.11 GO TO NEXT INSTRUCTION

GN

Use the **GN** command to set a temporary breakp
one following the current instruction. **GN** then
temporary breakpoint, the sequence of events is
already a breakpoint at the tempory breakpoint
than or equal to one or an error occurs.

GN is helpful when debugging modular code,
subroutine call as if it were a single instruction.

EXAMPLE The following section of code

```
CPU32Bug>MD 6000:4;DI<CR>
00006000   7003                      MOV
00006002   7201                      MOV
00006004   61000FFA                  BSR
00006008   2600                      MOV
CPU32Bug>
```

The following simple subroutine resides at addre

```
CPU32Bug>MD 7000:2;DI<CR>
00007000   D081                      ADD
00007002   4E75                      RTS
CPU32Bug>
```

Execute up to the BSR instruction.

```
CPU32Bug>RM PC<CR>
PC    =00003000 ? 6000.<CR>
CPU32Bug>GT 6004<CR>
Effective address: 00006004
Effective address: 00006000
At Breakpoint
PC    =00006004     SR    =2700=TR:OFF_S_
SFC   =0=F0         DFC   =0=F0         US
D0    =00000003     D1    =00000001     D2
D4    =00000000     D5    =00000000     D6
A0    =00000000     A1    =00000000     A2
A4    =00000000     A5    =00000000     A6
00006004   61000FFA              BSR.W
CPU32Bug>
```

3.13 GO TO TEMPORARY BREAKP

GT <addr>[:<count>]

Use the **GT** command to set a temporary breakpo be specified with the temporary breakpoint. C previously set breakpoints are enabled. The breakpoint with 0 count is encountered.

After setting the temporary breakpoint, the se command. At this point control is returned to CP

- Executing the .RETURN SYSCALL
- Press the ABORT switch or RESET s
- Encountering a breakpoint with 0 cou
- Generation of an unexpected exceptic

EXAMPLE The following program reside

```
CPU32Bug>MD 4000;DI<CR>
00004000   2200                     MOV
00004002   4282                     CLR
00004004   D401                     ADD
00004006   E289                     LSR
00004008   66FA                     BNE
0000400A   E20A                     LSR
0000400C   55C2                     SCS
0000400E   60FE                     BRA
CPU32Bug>RM D0<CR>
```

Initialize D0 and set a breakpoint:

```
D0   =00000000 ? 52A9C.<CR>
CPU32Bug>BR 400E<CR>
BREAKFOINTS
0000400E
CPU32Bug>
```

Set PC to beginning of program, set temporary b

```
CPU32Bug>RM PC<CR>
PC   =0000400E ? 4000.<CR>
CPU32Bug>
```


3.12 GO EXECUTE USER PROGRAM

GO [<addr>]

Use the **GO** command (alias **G**) to initiate target are enabled. If an address is specified, it is place PC address.

The sequence of events is:

1. An address is specified and loaded int
2. If a breakpoint is set at the target PC (executed in trace mode)
3. All breakpoints are inserted in the targ
4. Target code execution resumes at the

There are several methods for returning control t

- Execute the .RETURN TRAP #15 fur
- Press the ABORT switch or RESET s
- Encountering a breakpoint with 0 cou
- Generation of an unexpected exceptic

EXAMPLE The following program reside

```
CPU32Bug>MD 4000;DI<CR>
00004000   2200                     MOV
00004002   4282                     CLR
00004004   D401                     ADD
00004006   E289                     LSR
00004008   66FA                     BNE
0000400A   E20A                     LSR
0000400C   55C2                     SCS
0000400E   60FE                     BRA
CPU32Bug>RM D0<CR>
```

Initialize D0, set breakpoints, and start target pro

```
D0   =00000000 ? 52A9C.<CR>
CPU32Bug>BR 4000,400E<CR>
BREAKPOINTS
00004000            0000400E
CPU32Bug>GO 4000<CR>
Effective address:  00004000
At Breakpoint
PC    =0000400E     SR    =2711=TR:OFF_S_
SFC   =5=SD         DFC   =5=SD         US
D0    =00052A9C     D1    =00000000     D2
D4    =00000000     D5    =00000000     D6
A0    =00000000     A1    =00000000     A2
A4    =00000000     A5    =00000000     A6
0000400E   60FE                     BRA.B
```

Note that in this case breakpoints are inserted aft
breakpoint is not taken.

Continue target program execution.

```
CPU32Bug>G<CR>
Effective address:  0000400E
At Breakpoint
PC    =0000400E     SR    =2711=TR:OFF_S_
SFC   =5=SD         DFC   =5=SD         US
D0    =00052A9C     D1    =00000000     D2
D4    =00000000     D5    =00000000     D6
A0    =00000000     A1    =00000000     A2
A4    =00000000     A5    =00000000     A6
0000400E   60FE                     BRA.B
```

Remove breakpoints and restart target code.

```
CPU32Bug>NOBR<CR>
BREAKPOINTS
CPU32Bug>GO 4000<CR>
Effective address: 00004000
```


Press the **ABORT** pushbutton on the platform bo

```
Exception: ABORT
PC    =0000400E     SR    =2711=TR:OFF_S_
SFC   =5=SD         DFC   =5=SD         US
D0    =00052A9C     D1    =00000000     D2
D4    =00000000     D5    =00000000     D6
A0    =00000000     A1    =00000000     A2
A4    =00000000     A5    =00000000     A6
0000400E   60FE                     BRA.B
```

```
NOMAL        Disable Macro Expansion Lis
MD           Memory Display
MM           Memory Modify
M            "Alias" for previous comman
MS           Memory Set
OF           Offset Registers
PA           Printer Attach
NOPA         Printer Detach
PF           Port Format
RD           Register Display
RESET        Warm/Cold Reset
RM           Register Modify
RS           Register Set
SD           Switch Directory
T            Trace Instruction
TC           Trace on Change of Flow
TM           Transparent Mode
TT           Trace to Temporary Breakpoi
VE           Verify S-Records
CPU32Bug>
```

To display the command TC, enter:

```
CPU32Bug>HE TC<CR>
TC     Trace on Change of Flow
CPU32Bug>
```



```
CPU32Bug>GT 4006<CR>
Effective address: 00004006
Effective address: 00004000
At Breakpoint
PC    =00004006     SR    =2711=TR:OFF_S_
SFC   =0=F0         DFC   =0=F0         US
D0    =00052A9C     D1    =00000029    D2
D4    =00000000     D5    =00000000    D6
A0    =00000000     A1    =00000000    A2
A4    =00000000     A5    =00000000    A6
00004006   E289                        LSR.L
CPU32Bug>
```

Set another temporary breakpoint at $4002 and c

```
CPU32Bug>GT 4002<CR>
Effective address: 00004002
Effective address: 00004006
At Breakpoint
PC    =0000400E     SR    =2711=TR:OFF_S_
SFC   =0=F0         DFC   =0=F0         US
D0    =00052A9C     D1    =00000000    D2
04    =00000000     D5    =00000000    D6
A0    =00000000     A1    =00000000    A2
A4    =00000000     A5    =00000000    A6
0000400E   60FE                        BRA.B
```

Note that a breakpoint from the breakpoint
breakpoint.

3.14 HELP

HE [<command>]

HE is the CPU32Bug help facility. **HE** <**CR**>
plus any macro commands that have been defir
All CPU32Bug commands are in alphabetical or
Macro commands are displayed first, in the inve
HE command output fills the terminal screen,
"RETURN" to continue. Entering **he** <**comma**
title.

EXAMPLES

```
CPU32Bug>HE<CR>
BC       Block Compare
BF       Block Fill
BM       Block Move
BR       Breakpoint Insert
NOBR     Breakpoint Delete
BS       Block Search
BV       Block Verify
DC       Data Conversion and Expression
DU       Dump S-Records
GD       Go Direct (no breakpoints)
GN       Go and Stop after Next Instruc
GO       Go to Target Code
G        "Alias" for previous command
GT       Go and Insert Temporary Breakp
HE       Help Facility
LO       Load S-Records
MA       Macro Define/Display
NOMA     Macro Delete
MAE      Macro Edit
MAL      Enable Macro Expansion Listing
NOMAL    Disable Macro Expansion Listin
MD       Memory Display
MM       Memory Modify
M        "Alias" for previous command
MS       Memory Set
OF       Offset Registers
PA       Printer Attach
NOPA     Printer Detach
PF       Port Format
RD       Register Display
RESET    Warm/Cold Reset
RM       Register Modify
RS       Register Set
```



```
SD       Switch Directory
T        Trace Instruction
TC       Trace on Change of Flow
TM       Transparent Mode
TT       Trace to Temporary Breakpoint
VE       Verify S-Records
CPU32Bug>
```

To display the available commands in the diag
command and at the **CPU32Diag>** prompt enter

```
CPU32Bug>sd<CR>
CPU32Diag>he<CR>
DE          Display Errors
DP          Display Pass Count
LC          Loop-Continue Mode
LE          Loop-on-Error Mode
NV          NOn-Verbose Mode
MT          Memory Tests (Dir)
RL          Read Loop (Dir)
SE          Stop-on-Error Mode
SM          Modify Self-Test Mask
ST          Self Test Sequence
WL          Write Loop (Dir)
WR          Write/Read Loop (Dir)
ZE          Clear Error Counters
ZP          Zero Pass Count
BC          Block Compare
BF          Block Fill
BM          Block Move
BR          Breakpoint Insert
NOBR        Breakpoint Delete
BS          Block Search
BV          Block Verify
DC          Data Conversion and Express
DU          Dump S-Records
GD          Go Direct (no breakpoints)
GN          Go and Stop after Next Inst
GO          Go to Target Code
G           "Alias" for previous comman
GT          Go and Insert Temporary Bre
HE          Help Facility
LO          Load S-Records
MA          Macro Define/Display
NOMA        Macro Delete
MAE         Macro Edit
MAL         Enable Macro Expansion List
```

3.16 MACRO DEFINE/DISPLAY/DE

MA [<name>]

NOMA [<name>]

The <name> can be any combination of 1-8 alph

The **MA** command allows the user to define a c CPU32Bug primitive commands with optional p new <name> plus any arguments on the comm executed. This allows the user to design new **NOMA** command is used to delete either a singl

Entering **MA** without specifying a macro name macros and their definitions.

When **MA** is executed with the name of a cu displayed.

Line numbers are shown when displaying macr edit (**MAE**) command. If **MA** is executed wit definition, then the CPU32Bug enters the mac definition prompt "M=", enter a CPU32Bug com are not checked for syntax until the macro is exe only a carriage return (null line) in response to either be deleted and redefined or it can be edite no primitive CPU32Bug commands (i.e., no defi

Macro definitions are stored in a string pool of fi the definition mode, the offending string is disc LINE DISCARDED is printed and the user is re also happens if the string entered would cause th capacity of 511 characters. The only way to add to either edit or delete macros.

CPU32Bug commands contained in macros m time. Arguments are denoted in macro definitio numerial. As many as ten arguments are permitte by a zero would cause the first argument to characters.

3.15 LOAD S-RECORDS FROM HOS

LO [<port><del>][<addr>][;<X/-C/T>][=

Use the **LO** command to download a Motorola to the BCC. The **LO** command accepts serial memory.

The optional port number allows the user to spec the default is port 0.

The BCC default hardware configuration consis PFB. This limits the user to one host computer ru records, the user must escape out of the termina can not perform terminal emulation and send S-r terminal emulation mode, all status messages fro press <**CR**> twice after re-entering the termina status messages can now be sent.

The optional <addr> field allows the user to ent to the address contained in the address field of e in memory at a different location. The contents the S-record addresses (see **OF** command). If th number is omitted, enter a comma before the ad absolute addresses (i.e., "1000") should be unpredictable results. An address is allowed he support for function codes (see paragraph 2.5).

The optional text field, entered after the equal begins looking for S-records at the host port. Th to the host device. This text should NOT be de immediately following the equal sign and term operating full duplex, the string is echoed back user's terminal screen.

In order to accommodate host systems that echo a string is transmitted to and received from the command is sent to the host, **LO** looks for a line end of the echoed command. No data records ar system does not echo characters, **LO** continues are processed. In situations where the host system first record transferred by the host system be a h the **LF** after the header record serves to break **LO**

Other options:

-C Ignore checksum. A checksum
calculated as the S-record is rea
checksum is compared to the cl
compare fails, an error message
download. If this option is selecte

X Echo. As the S-records are read
terminal. Do not use this option w

T TRAP #15 code. This option caus
x = $0C ($4C4F200C). The AS
command; the code $0C indicates
passing and TRAP #15 disk su
program to select the appropriat
functions. Since some Motorol
CPU32Bug, they set a different co

The S-record format (refer to Appendix A) allov
the S-record-block termination record. The conte
any offset address) is put into the target PC. Thu
GO instead of **G** <**addr**> or **GO** <**addr**> to exec

If a non-hex character is encountered within the
record, preceeding the non-hex character, is disp
to point at the faulty character.

An error condition exists if the embedded-recor
calculated by CPU32Bug. An output message
from the address field of the record), the calcul
record. A copy of the record is also output. A
command to abort.

When a load is in progress, each data byte is v
memory location are compared to the data to det
reason the compare fails, then an output messag
stored, the data written, and the data read back c
causes the command to abort.

S-records are processed character-by-character. S
stored previous to the error is still in memory .

EXAMPLES Suppose a host computer was

```
                          *  Test Program
                          *
  65004000                      ORG
  
  65004000    7001              MOVEQ.L
  65004002    D088              ADD.L
  65004004    4A00              TST.B
  65004006    4E75              RTS
  END
****** TOTAL ERRORS    0--
****** TOTAL WARNINGS  0--
```

Then this program was converted into an S-recor

```
S00F00005445535453335337202001015E
S30D650040007001D0884A004E7577
S7056500400055
```

Load this file into BCC memory for execution at

```
CPU32Bug>LO -65000000<CR>
```

Enter the terminal emulator's escape key to retur
F4 for ProComm). A host command is then enter
BCC is connected (for MS-DOS based host comp
the BCC was connected to the com1 port).

After the file has been sent, the user then restart
based host computers, enter **EXIT** at the prompt)

Since the port number equals the current termina
that the download is complete and the terminal
messages.

```
<CR><CR>
CPU32Bug>
```

MAE Macr

EXAMPLES

```
CPU32Bug>MA<CR>
MACRO ABC
010 MD 3000
020 GO \0
CPU32Bug>
```

```
CPU32Bug>MAE ABC 15 RD<CR>
MACRO ABC
010 MD 3000
020 RD
030 GO \0
CPU32Bug>
```

```
CPU32Bug>MAE ABC 10 MD 10+R0<CR>
MACRO ABC
010 MD 10+R0
020 RD
030 GO \0
CPU32Bug>
```

```
CPU32Bug>MAE ABC 30<CR>
MACRO ABC
010 MD 10+R0
020 RD
CPU32Bug>
```


MA Macro Defi
NOMA Macro

The second argument is used whenever the seque the debugger command line would execute the m 1, and ;B replacing "\0", "\1", and "\2", respectiv

To delete a macro, execute **NOMA** followed without specifying a macro name deletes all ma name that does not have a definition, an error me

EXAMPLES

```
CPU32Bug>MA ABC<CR>                      Def
M=MD 3000
M=GO \0
M=<CR>
CPU32Bug>
```

```
CPU32Bug>MA DASM<CR>                     Def
M=MD \0:5;DI
M=<CR>
CPU32Bug>
```

```
CPU32Bug>MA<CR>                          List
MACRO ABC
010 MD 3000
020 GO \0
MACRO DIS
010 MD \0:5;DI
CPU32Bug>
```

```
CPU32Bug>DASM 427C<CR>                   Exe
0000427C    48E78080                     MOV
00004280    4280                         CLR
00004282    1018                         MOV
00004284    5340                         SUB
00004286    12D8                         MOV
CPU32Bug>
```

```
CPU32Bug>MA ABC<CR>                      List
MACRO ABC
010 MD 3000
020 GO \0
CPU32Bug>
```

```
CPU32Bug>NOMA DASM<CR>                   Del
CPU32Bug>
```

```
CPU32Bug>MA ASM<CR>
M=MM \0;DI
M=<CR>
CPU32Bug>
```

Def

```
CPU32Bug>MA<CR>
MACRO ABC
010 MD 3000
020 GO \0
MACRO ASM
010 MD \0;DI
CPU32Bug>
```

List

```
CPU32Bug>NOMA<CR>
CPU32Bug>
```

Del

```
CPU32Bug>MA<CR>
NO MACROS DEFINED
CPU32Bug>
```

List

3.17 MACRO EDIT

MAE <name><del><line#><del>[<string

Where:

<name>	any combination of 1-8 al
<line#>	line number in range 1-99
<string>	replacement line to be ins

The **MAE** command permits modification of th
line oriented and supports the following actions:

To insert a line, specify a line number between t
inserted between. The text of the new line to be
line following the line number.

To replace a line, specify its line number and en
the command line.

A line is deleted if its line number is specified an

Attempting to delete a nonexistent line results in
permit deletion of a line if the macro consists of
macro. To define new macros, use **MA**; the **MA**
macros.

Line numbers serve one purpose: specifying the l
editing function. After the editing is complete, th
line numbers.

3.20 MEMORY MODIFY

MM <addr>[;[[B|W|L][A][N]]|[DI]]

Use the **MM** command (alias **M**) to examine a
following data types:

Integer Data Type

B – Byte
W – Word
L – Longword

The default data type is word. The **MM** comma
memory at the specified address and prompts th
enter new data for the memory location, followe
leaves the contents unaltered. That memory loc
opened.

The user may also enter one of several step c
writing new data. Enter one of the following s
execution:

V or v	The next successive men initializes whenever **MM** by entering one of the othe
^	**MM** backs up and opens t
=	**MM** re-opens the same m registers or memory locati
.	Terminates **MM** command

The N option of the **MM** command disables th
forces alternate location accesses only, i.e. skip a
use.

NO

If the address location requested
offset register is non-zero and ha
the offset (**OF**) command.

3.18 MACRO EXPANSION LISTING

MAL
NOMAL

The **MAL** command allows the user to view exp
especially useful when errors result, as the line w

The **NOMAL** command is used to suppress the l

The use of **MAL** and **NOMAL** is a convenienc
function of the macros.

MD Memory

3.19 MEMORY DISPLAY

MD[S] <addr>[:<count>|<addr>][; [B|W|

Use the **MD** command to display the contents following data types:

Integer Data Type

B – Byte

W – Word

L – Longword

The default data type is word (W). Integer da ASCII. The DI option enables the resident MCU selected.

The optional count argument in the **MD** comr displayed, or the number of disassembled instı selected. The default is 8 if no value for <count> if the S (sector) modifier is used. After the com to re-execute the command and display the sam address.

EXAMPLES

```
CPU32Bug>md C000<CR>
0000C000 2800 1942 2900 1942  2800 1842

CPU32Bug><CR>
0000C010 FC20 0050 ED07 9F61  FF00 000A
```

Assume the following processor state: A2=0000

```
CPU32Bug>md (a2,d5):&19;b<CR>
00003627 4F82 00C5 9B10 337A  DF01 6C3D
00003637 31AB 80
CPU32Bug>
```

```
CPU32Bug>md 5008;di<CR>
00005008    46FC2700            MOV
0000500C    61FF0000023E        BSR
00005012    4E7AD801            MOV
00005016    41ED7FFC            LEA
0000501A    5888                ADD
0000501C    2E48                MOV
0000501E    2C48                MOV
00005020    13C7FFFB003A        MOV
CPU32Bug>
```

NO

If the address location requested offset register is non-zero and ha the offset (OF) command.

Offset register rules:

- At power-up and cold-start reset, R7
  have both base and top addresses pres
- R7 always has both base and top addr
- Any offset register can be set as the a
- The automatic register is always add
  CPU32Bug command where an offse
  the **OF** command itself). To enter an
  i.e. +R7.
- The register commands (**RD**, **RM**) d
  counter is always displayed/entered
  command does use the automatic regi
- There is always an automatic registe
  set R7 as the automatic register. This

EXAMPLES Display offset registers. Show

```
CPU32Bug>OF<CR>
R0 = 00000000 00000000 R1 = 00000000 00
R2 = 00000000 00000000 R3 = 00000000 00
R4 = 00000000 00000000 R5 = 00000000 00
R6 = 00000000 00000000 R7*= 00000000 00
```

Modify offset registers.

```
CPU32Bug>OF R0<CR>
R0 = 00000000 00000000? 5000 50FF<CR>
R1 = 00000000 00000000? 5100:200^<CR>
R0 = 00020000 000200FF? <CR>
R6 = 00000000 00000000? .<CR>
```

Display location $5000. Shows base and top valu

```
CPU32Bug>M 5000;DI<CR>
00000+R0 41F95445 5354        LEA.L  ($
CPU32Bug>M R0;DI <CR>
00000+R0 41F95445 5354        LEA.L  ($
CPU32Bug>
```


EXAMPLES

```
CPU32Bug>MM 3100<CR>
00003100 1234?<CR>
00003102 5678? 4321<CR>
00003104 9ABC? 8765^<CR>
00003102 4321?<CR>
00003100 1234? abcd.<CR>
```

```
CPU32Bug>MM 3001;LA<CR>
00003001 CD432187?<CR>
00003009 00068010? 68010+10=<CR>
00003009 00068020?<CR>
00003009 00068020? .<CR>
```

```
CPU32Bug>MM 4000<CR>
00004000 0000? 'A'<CR>
00004002 0000? 'B'<CR>
00004004 0000? 'CD'<CR>
00004006 0000? 'EFG'<CR>

00004008 0000? .<CR>
CPU32Bug>MD 4000<CR>
00004008 0041 0042 4344 4647 0000 0000
CPU32Bug>
```

The DI option activates the one-line assembler/di
selected. The contents of the specified memory
user prompted for an input with a question mark

- Enter <CR> – This closes the presen
  next instruction. The instruction is un
- Enter a new source instruction follo
  assemble the new instruction and gen
- Enter <CR> – This closes the present

If a new source line is entered (second option al
the new source line.

If an error is found during assembly, the car
followed by an error message. The accessed loca

Refer to Chapter 4 for additional information abo

3.21 MEMORY SET

MS <addr>{hexadecimal number}/{'stri

Use the **MS** command to write data to memory
not size specific, so they can contain any numb
size). If an odd number of digits is entered, the l
unchanged.

ASCII strings are entered by enclosing them in s
of the string, enter two consecutive quotes.

EXAMPLE Memory is initially cleared:

```
CPU32Bug>ms 25000 0123456789abcDEF 'Thi
CPU32Bug>md 25000:10;w<CR>
00025000 0123 4567 89AB CDEF  5468 6973
00025010 2733 3332 4275 6727  2345 6000
CPU32Bug>
```

NO

If the address location requeste
offset register is non-zero and ha
the offset (**OF**) command.

The **MS** command stores all data
should not be used on any locat
only, such as the MC68332 TP
requiring word accessing, use the
with the **;W** or **;L** option.

3.22 OFFSET REGISTERS DISPLAY

OF [Rn[;A]]

The **OF** command allows the user to access and
These registers are used to simplify the debu
modules (refer to offset registers in paragraph 2.1

There are 8 offset registers (R0 through R7), bu
base and top addresses of R7 is always set to 0.
selecting R7 as the automatic register.

Each offset register has two values: base and top
the range declared by the offset register. The to
When entering the base and top, the user ma
address/count format. When specifying a count t
omitted from the range, then a top address of $
equal or exceed the base address. Wrap-around is

Command usage:

OF	Display all offset registe automatic register.
OF Rn	Display/modify Rn. Scroll the MM command.
OF Rn;A	Display/modify Rn and s register is added to the except if an offset registe indicates which register is

Range entry:

Ranges are entered in three formats; base
addresses, and base address followed by
described in the **MM** (memory modify) c

Range syntax:

[<base address> [<del> <top addr

or

[<base address> [: <byte count>

(the next response demonstrates reversing the pr

```
XON/XOFF protocol [Y,N] = Y? ^ <CR>
Stop Bits [1,2] = 2? .<CR>

OK to proceed (y/n)? Y
CPU32Bug>
```

3.24.3 Port Format Parameters

The port format parameters are:

- Port base address – When assigning allows the user to adjust the base Entering no value selects the default a
- Baud rate – Select the baud rate: 110
- Parity type – Set parity: even (E), odd
- Character width – Select 5-, 6-, 7-, or
- Number of stop bits – Only 1 and 2 st
- Automatic software handshake – Cur XON/XOFF characters sent to the de to cease transmission until a XON cl utilize FIFO buffering, therefore, non
- Software handshake character values may be defined as any 8-bit value. AS accepted.

NO

Not all combinations of parity ty are supported for the BCC "SCI details.

Set R0 as the automatic register.

```
CPU32Bug>OF R0;A<CR>
R0*=00005000 000050FF? .<CR>
```

Display location 0 relative to the default offset re

```
CPU32Bug>M 0;DI<CR>
00000+R0 41F95445 5354
CPU32Bug>
```

Display absolute location 0, override the automat

```
CPU32Bug>M 0+R7;DI <CR>
00000000 FFF8
CPU32Bug>
```

3.23 PRINTER ATTACH/DETACH

PA [<port>]

NOPA [<port>]

PA attach a printer to a specified port. **NOPA** de printer is attached, everything appearing on th printer. If no port is specified when executing attached printers. The port number must be in the

If the port number specified is not currently assi attempted on a printer that is not currently attac (port format) command to configure the port bef

RECOVERING FROM A "HUNG"PRINTER: (bus errors, abort, etc). If **PA** is executed using jam occurs, press the RESET switch on the M68 printer.

<u>**EXAMPLES**</u>

<u>**CONSOLE DISPLAY:**</u>

```
CPU32Bug>PA <CR>
(attaching port 1 by default)

CPU32Bug>HE NOPA <CR>
NOPA      Printer detach

CPU32Bug>NOPA <CR>
(detach all attached printers)
CPU32Bug>
```


3.24 PORT FORMAT

PF [<port>]

Use the **PF** command to display and change t display a list of the current port assignments, co and configure a new port. The configuration pro or memory (**RM** and **MM** commands). An i hardware, the user must explicitly direct **PF** to p

Only eight ports are assigned at any given time. $1F.

3.24.1 List Current Port Assignments

Executing **PF** without specifying a port number l

<u>**EXAMPLE**</u>

```
CPU32Bug>PF <CR>
Current port assignments:  (Port #: Boa
00: BCC, "SCI"
CPU32Bug>
```

3.24.2 Port Configuration

Use **PF** to primarily change baud rates, stop bits port number to assign and configure port pa Assignment.

When **PF** is executed with the number of a pr entered immediately. To exit from the interacti new value/setting. While in the interactive mode (memory modify) command are supported.

<u>**EXAMPLE**</u> Change number of stop bits o

```
CPU32Bug>PF 0 <CR>
Baud rate [110,300,600,1200,2400,4800,9
Even, Odd, or No Parity [E,O,N] = N? <C
Char Width [5,6,7,8] = 8? <CR>
Stop bits [1,2] = 1? 2<CR>
```

EXAMPLES

```
CPU32Bug>rd<CR>
PC   =00003000    SR    =2700=TR:OFF_S_
SFC  =0=F0        DFC   =0=F0         US
D0   =00000000    D1    =00000000     D2
D4   =00000000    D5    =00000000     D6
A0   =00000000    A1    =00000000     A2
A4   =00000000    A5    =00000000     A6
00003000  424F                    DC.W
CPU32Bug>
```

NOT

An asterisk following a stack p
stack pointer. To facilitate readin
mnemonic portion. These mnemo

Trace Bits — The trace bits (T0, T1)
displayed by the mnemon
should not modify these bi

T1	T0	Mnemonic
0	0	TR:OFF
0	1	TR:CHG
1	0	TR:ALL
1	1	TR:INV

S Bits — The bit name (S) appears
period (.) indicates it is cle

Interrupt Mask — A number from 0 to 7 indi

Condition Codes — The bit name (X, N, Z, V,
a period (.) indicates it is c

3.24.4 New Port Assignment

PF supports a set of drivers for a number of diffe
a previously unassigned port number, execute the
then printed to indicate that the port is unassig
serial communication device. Pressing RETURN
and ports. Once the name of the board is entered
the port name is entered, **PF** prompts the user thr

Once a valid port is specified, default parameters
is one of these default parameters. Before enterin
allowed to change the port base address. Press **R**

If the configuration of the new port is not
configuration mode. Refer to paragraph 3.20.2 r
port has a fixed configuration, then **PF** issues the

The user must enter the letter ”Y” at the ”OK to p
hardware. Pressing **BREAK** any time prior to t
prompt leaves the port unassigned. This is only t

EXAMPLE Assigning port 1.

```
CPU32Bug>PF 1<CR>
Logical unit $01 unassigned
Name of board?<CR>
Boards and ports supported:
BCC: SCI
MC68681: A, B
Name of board? mc68681<CR>
Name of port? a<CR>
Port base address = $FFFFE000?<CR>
Baud rate [110, 300, 600, 1200, 2400, 4
OK to proceed (Y/N)? n
CPU32Bug>
```

3.25 REGISTER DISPLAY

RD {[+|-|=][<dname>][/]}{[+|-|=][<reg1

Use the **RD** command to display the target stat target program (refer to the **GO** command). disassembled and displayed. Internally, a regist when **RD** <**CR**> is executed. At reset time, this Change this register mask with the **RD** com capability to enable or disable the display of any showing only the registers of interest, minimizin

The arguments are:

+	Add a device or register ra
-	Remove a device or regist names. In which case it in
=	Set a device or register ran
/	Use this delimiter betwee
<reg1>	Indicates the first register
<reg2>	Indicates the last register i
<dname>	Indicates a device name. registers for:

MPU Microproces

Observe the following when specifying any argu

- The qualifier is applied to the next reg
- If no qualifier is specified, a + qualifi
- All device names should precede regi
- The command line arguments are p processed after parsing, thus, the s organized has an impact on the result
- When specifying a register range, < same class, i.e. D0 - A7.
- The register mask used by **RD** is a including the trace and breakpoint exc

The MPU registers in ordering sequence are:

Number of registers	
10	System Registers
8	Data Registers
8	Address Registers

3.28 REGISTER SET

RS <reg>[<exp>][;A]

Use the **RS** command to display or change a s value is always added to <exp> unless overridde the **OF** (offset register) command.

The ;A option is only valid when <reg> is an off <reg> as the automatic register. If R7 is specifie See the **OF** (offset register) command.

EXAMPLES

```
CPU32Bug>RS PC 40*1000+4<CR>
PC   =00040004
CPU32Bug>
```

```
CPU32Bug>OF R4;A<CR>
R4*00000000 00000000? 4000 4FFF<CR>
CPU32Bug>RS PC 124<CR>
PC   =00004124
CPU32Bug>RS A4 32A<CR>
A4   =0000432A
CPU32Bug>RS A5 400+R7<CR>
A5   =00000400

CPU32Bug>
```


The source and destination function code re mnemonic:

Function Code	M
0	
1	
2	
3	
4	
5	
6	
7	

To set the display to D6 and A3 only.

```
CPU32Bug>RD =D6/A3<CR>
D6   =00000000 A3   =00000000
00003000 4AFC              ILLEGAL
CPU32Bug>
```

Note that the above sequence sets the display to I

To restore all the MPU registers.

```
CPU32Bug>rd +mpu<CR>
PC   =00003000      SR   =2700=TR:OFF_S_
SFC  =0=F0          DFC  =0=F0
D0   =00000000      D1   =00000000
D4   =00000000      D5   =00000000
A0   =00000000      A1   =00000000
A4   =00000000      A5   =00000000
00003000 4AFC              ILLEGAL
CPU32Bug>
```

Note that an equivalent command is ’’RD +PC-A

3.26 COLD/WARM RESET

RESET

Use the **RESET** command to specify the rese
detected by the processor. Press the RESET
generate a reset exception.

Two **RESET** levels are available:

COLD This is the standard mode
this mode all the static
executed.

WARM In this mode all the static
occurs. This is convenien
the target register state, an

EXAMPLE

```
CPU32Bug>RESET<CR>
Cold/Warm Start = C (C/W)? W<CR>
CPU32Bug>

CPU32Bug Debugger/Diagnostics - Version
  (C) Copyright 1991 by Motorola Inc.
Warm Start

CPU32Bug>
```


3.27 REGISTER MODIFY

RM <reg>

Use the **RM** command to display and change the
essentially the same way as the **MM** command,
control the display/change session. Refer to the M

EXAMPLES

```
CPU32Bug>RM D4<CR>
D5   =12345678? ABCDEF^<CR>
D4   =00000000? 3000.<CR>
CPU32Bug>
```

```
CPU32Bug>rm sfc<CR>
SFC  =7=CS  ? 1=<CR>
SFC  =1=UD  ? .<CR>
CPU32Bug>
```

Trace the next two instructions:

```
CPU32Bug>T 2<CR>
PC   =00007006      SR   =2700=TR:OFF_S_
SFC  =0=F0          DFC  =0=F0
D0   =0008F41C      D1   =0008F41C
D4   =00000000      D5   =00000000
A0   =00000000      A1   =00000000
A4   =00000000      A5   =00000000
00007006 E289             LSR.L   #$1
PC   =00007008      SR   =2700=TR:OFF_S_
SFC  =0=F0          DFC  =0=F0
D0   =0008F41C      D1   =00047A0E
D4   =00000000      D5   =00000000
A0   =00000000      A1   =00000000
A4   =00000000      A5   =00000000
00007008 66FA             BNE.B   $70
CPU32Bug>
```


3.29 SWITCH DIRECTORIES

SD

Use the **SD** command to toggle between the debu

Use the **HE** (Help) command to list the current d

Directory structure allows access to the debu diagnostic commands are only available from the

EXAMPLES

```
CPU32Bug>SD<CR>
CPU32Diag>
```

```
CPU32Diag>SD<CR>
CPU32Bug>
```

3.30 TRACE

T [<count>]

Use the **T** command to execute one instructi
execution. **T** starts tracing at the address in the
number of instructions to be traced before ret
default is 1. As each instruction is traced, a regis

During tracing, breakpoints in ROM or write pr
for all trace commands which allow the use of b
Control is returned to CPU32Bug if a breakpoint

Trace functions are implemented with the trace
Do not modify trace bits (T0, T1) while using t
are implemented using the hardware trace bits i
trace mode, breakpoints are monitored and the
instruction with breakpoint is traced. This allow
trace mode.

EXAMPLE The following program reside

```
CPU32Bug>MD 7000;DI<CR>
00007000 2200                   MOVE.L
00007002 4282                   CLR.L
00007004 D401                   ADD.B
00007006 E289                   LSR.L
00007008 66FA                   BNE.B
0000700A E20A                   LSR.B
0000700C 55C2                   SCS.B
0000700E 60FE                   BRA.B
CPU32Bug>
```

Initialize PC and D0:

```
CPU32Bug>RM PC<CR>
PC   =00008000 ? 7000.<CR>
```

```
CPU32Bug>RM D0 <CR>
D0   =00000000 ? 8F4IC.<CR>
```


Display target registers and trace one instruction

```
CPU32Bug>RD<CR>
PC   =00007000      SR   =2700=TR:OFF_S_
SFC  =0=F0          DFC  =0=F0
D0   =0008F41C      D1   =00000000
D4   =00000000      D5   =00000000
A0   =00000000      A1   =00000000
A4   =00000000      A5   =00000000
00007000 2200             MOVE.L   D0,
```

```
CPU32Bug>T<CR>
PC   =00007002      SR   =2700=TR:OFF_S_
SFC  =0=F0          DFC  =0=F0
D0   =0008F41C      D1   =0008F41C
D4   =00000000      D5   =00000000
A0   =00000000      A1   =00000000
A4   =00000000      A5   =00000000
00007002 4282             CLR.L    D2
CPU32Bug>
```

Trace next instruction:

```
CPU32Bug><CR>
PC   =00007004      SR   =2704=TR:OFF_S_
SFC  =0=F0          DFC  =0=F0
D0   =0008F41C      D1   =0008F41C
D4   =00000000      D5   =00000000
A0   =00000000      A1   =00000000
A4   =00000000      A5   =00000000
00007004 D401             ADD.B    D1,
CPU32Bug>
```

3.33 TRACE TO TEMPORARY BRE

TT <addr>

Use the **TT** command to set a temporary bre
encountering a 0 count breakpoint. The tempora
to the **GT** command) and control is returned
address. As each instruction is traced, a register d

During tracing, breakpoints in ROM or write pro
for all trace commands which allow the use of br
breakpoint with 0 count is encountered. See the t

The trace functions are implemented with the tra
not modify trace bits (T0, T1) while using the tr
implemented using the hardware trace bits in the
mode, breakpoints are monitored and their
instruction with breakpoint is traced. This allow
trace mode.

EXAMPLE The following program reside

```
CPU32Bug>MD 7000;DI<CR>
00007000 2200                    MOVE.L
00007002 4282                    CLR.L
00007004 D401                    ADD.B
00007006 E289                    LSR.L
00007008 66FA                    BNE.B
0000700A E20A                    LSR.B
0000700C 55C2                    SCS.B
0000700E 60FE                    BRA.B
CPU32Bug>
```

Initialize PC and D0:

```
CPU32Bug>RM PC<CR>
PC    =00008000 ? 7000.<CR>
```

```
CPU32Bug>RM D0<CR>
D0    =00000000 ? 8F41C.<CR>
```


3.31 TRACE ON CHANGE OF CONT

TC [<count>]

Use the **TC** command to start execution at th
detection of an instruction that causes a change
etc. Execution is in real time until a change o
count field specifies the number of change of
control to CPU32Bug. The optional count field d
when a change of control flow occurs.

During tracing, breakpoints in ROM or write pro
for all trace commands which allow the use
recognizes a breakpoint only if it is at a chan
CPU32Bug if a breakpoint with 0 count is enc
details.

The trace functions are implemented with the
register. Do not modify the trace bits (T0, T1) wh
functions are implemented using the hardware
traced. During trace mode, breakpoints are mon
corresponding instruction with breakpoint is tra
but only in the trace mode.

EXAMPLE The following program reside

```
CPU32Bug>MD 7000;DI<CR>
00007000 2200                    MOV
00007002 4282                    CLR
00007004 D401                    ADD
00007006 E289                    LSR
00007008 66FA                    BNE
0000700A E20A                    LSR
0000700C 55C2                    SCS
0000700E 60FE                    BRA
CPU32Bug>
```

Initialize PC and D0:

```
CPU32Bug>RM PC <CR>
PC    =00008000 ? 7000.<CR>
```

```
CPU32Bug>RM D0 <CR>
D0    =00000000 ? 8F41C.<CR>
```

Trace on change of flow:

```
CPU32Bug>TC<CR>
00007008 66FA                   BNE.B  $70
PC   =00007004      SR   =2700=TR:OFF_S_
SFC  =0=F0          DFC  =0=F0
D0   =0008F41C      D1   =00047A0E
D4   =00000000      D5   =00000000
A0   =00000000      A1   =00000000
A4   =00000000      A5   =00000000
00007004 D401                   ADD.B  D1,
CPU32Bug>
```

Note that the above display also shows the chan

3.32 TRANSPARENT MODE

TM [<port>][<escape>]

The **TM** command connects the console serial p
to communicate with a host computer. A mess
character, i.e., the character used to exit the tra
until the escape character is received by the cons
to the host and at power up or reset is initialized

The optional port number allows the user to s
number is omitted the default is port 1. The port

Ports do not have to have the same baud rate, b
rate should be equal to or greater than the host
baud rates.

The optional escape argument allows the user
formats:

ascii code	:	$03	Set es
ascii character	:	’c	Set es
control character	:	^C	Set es

If the port number is omitted and the escape arg
escape argument with a comma to distinguish it f

EXAMPLES

```
CPU32Bug>TM<CR>
Escape character:  $01=^ A
<^A>
```

```
CPU32Bug>TM ^g<CR>
Escape character:  $07=^ G
<^G>
CPU32Bug>
```

Then converted into an S-Record file named TES

```
S00A0000544553542E4D58E2
S30D650040007001D0884A004E7577
S7056500400055
```

This file was downloaded into memory using "
may be examined in memory using the **MD** (mer

```
CPU32Bug>MD 4000:4;DI<CR>
00004000 7001           MOVEQ.L     #$1
00004002 D088           ADD.L       A0,
00004004 4A00           TST.B       D0
00004006 4E75           RTS
CPU32Bug>
```

To ensure the program has not been destroyed
verification.

```
CPU32Bug>VE -65000000<CR>
```

Enter the terminal emulator's escape key to retur
F4 for ProComm). A host command is then enter
BCC is connected (for MS-DOS based host comp
the BCC was connected to the com1 port).

After the file has been sent, the user then restart
based host computers, enter **EXIT** at the prompt)

Since the port number equals the current termina
that verification is complete and the terminal e
message.

```
<CR><CR>
Verify passes.
CPU32Bug>
```

The verification passes. The program stored in r
S-record file.

Trace to temporary breakpoint:

```
CPU32Bug>TT 7006<CR>
PC   =00007002      SR   =2700=TR:OFF_S_
SFC  =0=F0          DFC  =0=F0
D0   =0008F41C      D1   =0008F41C
D4   =00000000      D5   =00000000
A0   =00000000      A1   =00000000
A4   =00000000      A5   =00000000
00007002 4282               CLR.L   D2
PC   =00007004      SR   =2704=TR:OFF_S_
SFC  =0=F0          DFC  =0=F0
D0   =0008F41C      D1   =0008F41C
D4   =00000000      D5   =00000000
A0   =00000000      A1   =00000000
A4   =00000000      A5   =00000000
00007004 D401               ADD.B   D1,
At Breakpoint
PC   =00007006      SR   =2700=TR:OFF_S_
SFC  =0=F0          DFC  =0=F0
D0   =0008F41C      D1   =0008F41C
D4   =00000000      D5   =00000000
A0   =00000000      A1   =00000000
A4   =00000000      A5   =00000000
00007006 E289               LSR.L   #$1
CPU32Bug>
```

3.34 VERIFY S-RECORDS AGAINST

VE [<port>][<addr>][;<X/-C>][=<text>]

VE is identical to the **LO** command with the e
merely compared to the contents of memory.

The **VE** command accepts serial data from a hos
and compares it to data already in memory. If th
via information sent to the terminal screen.

The optional port number allows the user to s
number is omitted the default is port 0. The port

The BCC default hardware configuration consis
PFB. This limits the user to one host computer ru
records, the user must escape out of the termina
can not perform terminal emulation and send S-r
terminal emulation mode, all status messages fro
press <**CR**> twice after re-entering the termina
status messages can now be sent.

The optional <addr> field allows the user to ente
contained in the record address field. This cau
different locations than would normally occur. T
not added to the S-record addresses. If the addre
is omitted, precede the address with a comma to
addresses (i.e., "1000") should be entered, as oth
An address is allowed here rather than an offset (
(see paragraph 2.5).

The optional text field, entered after the equals
begins to look for S-records at the host port. Thi
device to initiate the download. Do not delimit
equals sign and terminates with a carriage retur
echoes back to the host port and appears on the u

Some host systems echo all received characters s
host one character at a time. After the entire co
character from the host signifying the end of the
until LF is received. If the host system does
character before data records are processed. For

host system does not echo characters that the f
header record. The header record is not used, bu
VE out of the loop so that data records are proce

Other **VE** options are:

-C option Ignore checksum. A check
is calculated as the S-recor
checksum is compared to t
if the compare fails, an err
selected, the comparison is

X option Echo. This option echoes
read in at the host port. Do

During a verify operation S-record data is com
address contained in the S-record address field (
then the non-comparing record is set aside until t
the screen. If three non-comparing records are e
then the command is aborted.

If a non-hex character is encountered within the d
is printed to the screen and CPU32Bug's error ha

An error condition exists if the embedded ch
checksum calculated by CPU32Bug. A message
(as obtained from the address field of the recor
read with the record. A copy of the record is a
command to abort.

EXAMPLES

This short program was developed on a host syst

```
                      * Test Program
                      *
   65004000                 ORG

   65004000 7001            MOV
   65004002 D088            ADD
   65004004 4A00            TST
   65004006 4E75            RTS
   END
****** TOTAL ERRORS      0--
****** TOTAL WARNINGS    0--
```

4.1.2 M68300 Family Resident Structured Ass

There are several major differences between the resident structured assembler. The resident asse entire program as a unit, while the CPU32Bug a individual unit. Due mainly to this basic fu capabilities are more restricted:

- Label and line numbers are not use locations in a program. The one-line therefore, cannot make the require definition located on a separate line.
- Source lines are not saved. In order machine code is disassembled and the
- Only two directives (DC.W and SYS
- No macro operation capability is incl
- No conditional assembly is used.
- No structured assembly is used.
- Several symbols recognized by the CPU32Bug assembler character set. symbols have multiple meaning to th These are:

  Asterisk (*) - Multiply

  Slash (/) - Divide or

  Ampersand (&) - And or de

Although functional differences exist between t true subset of the resident assembler. The CPU32 to the resident assembler except as described abo

4.2 SOURCE PROGRAM CODING

A source program is a sequence of source state predetermined tasks. Each source statement oc instruction, a DC.W directive, or a SYSCAL follows a consistent source line format.

Now change the program in memory and perform

```
CPU32Bug>M 4002<CR>
00004002 D088 ? D089.<CR>

CPU32Bug>VE -65000000<CR>
```

Enter the terminal emulator's escape key to retur F4 for ProComm). A host command is then enter BCC is connected (for MS-DOS based host com the BCC was connected to the com1 port).

After the file has been sent, the user then restart based host computers, enter **EXIT** at the prompt)

Since the port number equals the current termina that verification is complete and the terminal e message.

```
<CR><CR>

S30D65004000------88--------77
CPU32Bug>
```

The byte which was changed in memory does no record.

CHAP

ASSEMBLER/D

4.1 INTRODUCTION

Included as part of the CPU32Bug firmware is a
assembler is an interactive assembler/editor in
source line is translated into M68300 Family m
into memory as it is entered. In order to display a
and the instruction mnemonic and operands are d
are translated.

The CPU32Bug assembler is effectively a sub
assembler. It has some limitations as compared w
line numbers and labels; however, it is a power
code of the M68300 Family.

4.1.1 M68300 Family Assembly Language

M68300 Family assembly language is the symb
processing by the assembler. This language is a c

- Operations
  - M68300 Family machine-instr
  - Directives (pseudo-ops)
- Operators
- Special symbols

4.1.1.1 Machine-Instruction Operation Code

The part of the assembly language that provide
codes for the M68300 Family machine instru
Manual. Refer to that manual for any questions c

4.1.1.2 Directives

Normally, assembly language can contain m
auxiliary action. The CPU32Bug assembler r
constant) and SYSCALL. These two directive
CPU32Bug utility calls (refer to paragraphs 4.2.3

4.2.1.5 Character Set

The character set recognized by the CPU32Bug a

- The letters A through Z (uppercase ar
- The integers 0 through 9
- Arithmetic operators: +, -, *, /, <<, >
- Parentheses ( )
- Characters used as special prefixes:

  # (pound sign) specifies the im

  $ (dollar sign) specifies a hexa

  & (ampersand) specifies a deci

  @ (commercial at sign) specifi

  % (percent sign) specifies a bir

  ’ (apostrophe) specifies an AS(
- Five separating characters:

  Space

  . (period)

  / (slash)

  - (dash)
- The asterisk (*) character indicates cu

4.2.2 Addressing Modes

Effective address modes, combined with op performed by a given instruction. Effective add detail in the CPU32 Reference Manual.

4.2.1 Source Line Format

Each source statement is a combination of op numbers, labels and comments are not used.

4.2.1.1 Operation Field

Since there is no label field, the operation field also follow one or more spaces. Entries can cons

- Operation codes which correspond to
- Define constant directive (DC.W) def
- System call directive (SYSCALL) cal

The size of the data field affected by an instruct instructions and directives can operate on more t size code must be specified or a default size app need not be specified if only one data size is pe followed by a period (.) and the data size code. T

B	= Byte (8-bit data)
W	= Word (16-bit data; the usual def
L	= Longword (32-bit data)

When the instruction or directive does not have permitted.

<u>EXAMPLES</u> Legal

`LEA`	`(A0),A1`	Load the effectiv size is the defau
`ADD.B`	`(A0),D0`	Add the byte poi
`ADD`	`D1,D2`	Add the low ord default size code
`ADD.L`	`A3,D3`	Add the entire 32

<u>EXAMPLE</u> Illegal

`SUBA.B`	`#5,A1`	Illegal size spec instruction would of A1; byte opera

4.2.1.2 Operand Field

If present, the operand field follows the operatic
by at least one space. When two or more opera
them with a comma. In an instruction like ’AD
source effective address (<EA>) field, and the
<EA> field. Thus, the contents on D1 are adde
register D2. In the instruction ’MOVE D1,D2’, t
second subfield (D2) is the destination field. In
the format ’<opcode> <source>,<destination>’ a

4.2.1.3 Disassembled Source Line

The disassembled source line may not look ident
decides how to interpret the numbers used. If th
treated as a signed hexadecimal offset. Oth
hexadecimal.

<u>EXAMPLE</u>

```
MOVE.L      #1234,5678
MOVE.L      FFFFFFFC(A0),5678
```

disassembles to

```
00003000    21FC0000 12345678
00003008    21E8FFFC 5678
```

Also, for some instructions, there are two valid n
than one assembly language equivalent. When th
appear different from the originally entered code

BT is dissembled as BRA

DBRA is dissembled as DBF

NO

The assembler recognizes two for
instructions. The **BT** form (branc
opcode as the **BRA** instruction
branch always) and **DBF** (neve
mnemonics are different forms f
case, the assembler accepts both f

4.2.1.4 Mnemonics and Delimiters

The assembler recognizes all M68300 Family ins
binary, octal, decimal, and hexadecimal, with he

- Decimal values are preceded by an ar

  &12334

  -&987654321

- Hexadecimal values are preceded by

  $AFE5

One or more ASCII characters enclosed by sing
ASCII strings are right-justified and zero fill
immediate operands.

```
00003000    21FC0000 12345678
005000      0053
005002      223C41424344
005008      3536
```

The following register mnemonics are recognize

Pseudo R	
R0-R7	User Offset Registers.
Main Process	
PC	Program Counter - Used only in fo
SR	Status Register
CCR	Condition Codes Register (Lower
USP	User Stack Pointer
SSP	System Stack Pointer
VBR	Vector Base Register
SFC	Source Function Code Register
DFC	Destination Function Code Regist
D0-D7	Data Registers
A0-A7	Address Registers - Address regi pointer, that is, either USP or S register

<u>**EXAMPLES**</u>

```
00010022    04D2    DC.W    123
00010024    AAFE    DC.W    &A
00010026    4142    DC.W    'AE
00010028    5443    DC.W    'TB
0001002A    0043    DC.W    'C'
```

4.2.4 System Call Directive (SYSCALL)

This directive aids the user in making the TRAP
this directive is:

SYSCALL <function name>

For example, the following two pieces of code pr

```
        TRAP        #$F
        DC.W        0
    or
        SYSCALL     .INCHR
```

The CPU32Bug input default is hexadecimal, w
programming a CPU32Bug assembler TRAP fu
and let CPU32Bug make the conversion. Refer t
listing of all the functions provided.

4.3 ENTERING AND MODIFYING S

User programs are entered into memory using th
is entered in assembly language statements on a
as it is converted immediately upon entry into ma
the type of source line that can be entered.

Symbols and labels, other than the defined
assembler has no means of storing the associate
tables. This forces the programmer to use memor
use labels.

Also, editing is accomplished by retyping an e
moving a block of memory data to free up or del
the **BM** command).

Table 4-1 summarizes the CPU32Bug one-line a

Table 4-1. CPU32Bug Ass

Format	
Dn	Data register direct
An	Address register direct
(An)	Address register indirec
(An)+	Address register indirec
-(An)	Address register indirec
d(An)	Address register indirec
d(An,Xi)	Address register indirec
(bd,An,Xi)	Address register indirec
ADDR(PC)	Program counter indirec
ADDR(PC,Xi)	Program counter indirec
(ADDR,PC,Xi)	Program counter indirec
(xxxx).W	Absolute word address
(xxxx).L	Absolute long address
#xxxx	Immediate data

The user may use an expression in any numeric
has a built in expression evaluator that supports t

Binary numbers
Octal numbers
Decimal numbers
Hexadecimal numbers
String literals
Offset registers
Program counter

Allowed operators are:

Addition	+
Subtraction	-
Multiply	*
Divide	/
Shift left	<<
Shift right	>>
Bitwise or	!
Bitwise and	&

The order of evaluation is strictly left to right wi
others. The only exception is when the user
parentheses.

Possible points of confusion:

- Differentiate numbers and registers tc

  CLR D0 means CLR.W reg

  CLR $D0
  CLR 0D0
  CLR +D0
  CLR D0+0 all mean CLR.W n

- With the use of asterisk (*) to repre
  how does the assembler know when t

  For parsing algebraic expressions

  <OPERAND> <OPERAT

  with a possible left or right parent

  Given the above order, the as
  definition to use. For example:

***	Means
+	Means
2**	Means
*&&16	Means

When specifying operands, the user may skip
modes.

- Address register indirect with index, l
- Program counter indirect with index,

For the above modes, the rules for omission/skip

- The user may terminate the operand b

  EXAMPLE

  CLR ( ) or
  CLR (,,) is equivalent to
  CLR (0.N,ZA0,ZD0.W*1)

- The user may skip a field by stepping

  EXAMPLE

  CLR (D7) is equivalent to
  CLR ($D7,ZA0,ZD0.W*1)
  but
  CLR (,,D7) is equivalent to
  CLR (0.N,ZA0,D7.W*1)

- If the user does not specify the base
  the register number, it indicates that r
- If the user does not specify the index
- Any unspecified displacements are de

4.2.3 Define Constant Directive (DC.W)

The format for the DC.W directive is:

DC.W <operand >

This directive defines a constant in memory. Th
value) which can contain the actual value (deci
operand can be an expression which is assigned
is aligned on a word boundary if word (.W) size i
characters are enclosed inside single quotes mark
byte of memory with the eighth bit (MSB) alwa
byte is right justified. A maximum of two AS
directive.

It is necessary to create an equate file with the ro download the archive file C32SCALL.ARC fr (BBS). For more information on the FR M68xxxEVx/L2.

When using the CPU32Bug one-line assemble equates are pre-defined. Input: SYSCALL, space

EXAMPLE

```
CPU32Bug>M 3000;DI<CR>
0000 3000 00000000         ORI.B
0000 3000 4E3F0022         SYSCAL
0000 3004 00000000         ORI.B
CPU32Bug>
```

5.1.2 Input/Output String Formats

Within the context of the TRAP #15 handler are

Pointer/Pointer Format	The string is d pointer to the la
Pointer/Count Format	The string is contains the co the string itself
Line Format	A line is define line feed.

5.2 SYSTEM CALL ROUTINES

Table 5-1 summarizes the TRAP #15 functio description of the available system calls.

4.3.1 Executing the Assembler/Disassemble

The assembler/disassembler is actuated using th **MD** (Memory Display) commands:

MM <ADDR >;DI

where

<**CR**> sequences to next i

.<**CR**> exits command

and

MD[S] <ADDR>[:<count>I<ADD

Use the **MM** (;DI option) to enter and modify memory contents at the specified location are d line can be entered if desired.

The disassembled line is either an M68300 directive. If the disassembler recognizes a valid f If the disassembler does not recognizes a valid DC.W $XXXX (always hex) is returned. B instructions, a word of data interpreted as a valid

4.3.2 Entering a Source Line

Enter a new source line immediately following in paragraph 4.2.1.

```
CPU32Bug>MM 6000;DI <CR>
00006000                2600  MOV
```

When a line is terminated with a carriage return screen, the new line is assembled and displa disassembled and displayed:

```
CPU32Bug>MM 6000;DI<CR>
00006000                528B  ADD
00006002                4282  CLR
```

Another program line can now be entered. Progr have been entered. A period (.) is used to exit th assembly, the assembler displays the line unasser accessed is redisplayed:

```
CPU32Bug>MM 6000;di <CR>
00006000                528B  ADD
LEA.L 5(A0,D8),A4
BAD COMBINATION OF COMMAND, OPERA
00006000                528B  ADD
```

4.3.3 Entering Branch and Jump Addresses

When entering a source line containing a branch
the offset to the branch's destination in the instr
the assembler. The user must append the appropr

To reference a current location in an operand exp

EXAMPLES

```
0000D000                  6000BF68
0000D000                  60FE
0000D000                  4EF90000
0000D000                  4EF00130
```

In the case of forward branches or jumps, th
unknown as the program is being entered. The us
order to reserve space. After the actual addres
instruction can be re-entered using the correct va
''S'' and ''L''.

4.3.4 Assembler Output/Program Listings

Use the **MD** (Memory Display) command with t
The **MD** command requires the starting address
in the command line. When the ;DI option i
instructions disassembled and displayed is equal

Note again, that the listing may not correspond e
paragraph 4.2.1.3, the disassembler displays in si
offset of an address register; all other numbers ar

CHAP

SYSTEM

5.1 INTRODUCTION

This chapter describes the CPU32Bug TRAP #1
programs. System calls access selected func
including input and output routines. TRAP #15
end of a user program (refer to the .RETURN fu

In the descriptions of some input and output func
or the default output port. After power-up or rese
BCC terminal port).

5.1.1 Executing System Calls Through TRA

To execute a system call from a user program
source program. The code corresponding to the p
following the TRAP opcode, as shown in the foll

Format in user program:

TRAP #15	Sys
DC.W $xxxx	Rou

In some of the examples shown in the followin
the Motorola Macro Assembler (M68MASM)
automatically assembles the TRAP #15 call fo
code. The SYSCALL macro is:

```
SYSCALL    MACRO
           TRAP     #15
           DC.W     \1
           ENDM
```

The CPU32Bug input default is hexadecimal, w
programming a CPU32Bug assembler TRAP fu
make the conversion.

Using the SYSCALL macro, the system call app

```
SYSCALL    <routine name>
```

.CHANGEV Parse Value, As

If the above code was called with a syscall routi format and POINT contained 2 (longwords), th POINT would contain 4 (pointing to first charac buffer start address (not the address of the first process. In this case, a value of 2 in POINT indi character to be processed. After calling .CHANG

```
COUNT = 3
```

If the above code was called again, nothing cou be issued. For example, if the string 5 is entered

```
COUNT = 3? 5<CR>
COUNT = 5
```

If in the previous example nothing had been ente value.

```
COUNT = 3? <CR>
COUNT = 3
```


Table 5-1. CPU32Bug

Function	Trap Code	
.BINDEC	$0064	Convert binar
.CHANGEV	$0067	Parse value
.CHKBRK	$0005	Check for brea
.DELAY	$0043	Timer delay fu
.DIVU32	$006A	Divide two 32
.ERASLN	$0027	Erase line
.INCHR	$0000	Input characte
.INLN	$0002	Input line (poi
.INSTAT	$0001	Input serial po
.MULU32	$0069	Multiply two 3
.OUTCHR	$0020	Output charac
.OUTLN	$0022	Output line (p
.OUTSTR	$0021	Output string
.PCRLF	$0026	Output carriag
.READLN	$0004	Input line (poi
.READSTR	$0003	Input string (p
.RETURN	$0063	Return to CPU
.SNDBRK	$0029	Send break
.STRCMP	$0068	Compare two
.TM_INI	$0040	Timer initializ
.TM_RD	$0042	Read timer
.TM_STR0	$0041	Start timer at
.WRITD	$0028	Output string
.WRITDLN	$0025	Output line wi
.WRITE	$0023	Output string
.WRITELN	$0024	Output line (p

.BINDEC Calculate BCD Equivalent

5.2.1 Calculate BCD Equivalent Specified E

SYSCALL .BINDEC
TRAP CODE: $0064

This function takes a 32-bit unsigned binary
(Binary Coded Decimal Number).

Entry Conditions:

SP ==> Argument: Hex nu
Space for result

Exit Conditions:

SP ==> Decimal number

EXAMPLE

```
SUBQ.L      #8,A7          Allocate
MOVE.L      D0,-(A7)       Load hex
SYSCALL     .BINDEC        Call .BIN
MOVEM.L     (A7)+,D1/D2    Load resu
```


.CHANGEV Parse Value, As

5.2.2 Parse Value, Assign to Variable

SYSCALL .CHANGEV
TRAP CODE: $0067

Parse a value in the user specified buffer. If
prompted for a new value, otherwise update the
The new value is displayed and assigned to the v

Entry Conditions:

SP ==> Address of 32-bit o
Address of user's b
Address of 32-bit i
Address of string to

Exit Conditions:

SP ==> Top of stack

EXAMPLE

```
PROMPT      DC.B       $14,'COUNT = |1
GETCOUNT    PEA        PROMPT(PC)
            PEA        COUNT
            PEA        BUFFER
            PEA        POINT
            SYSCALL    .CHANGEV
            RTS
```

5.2.6 Erase Line

SYSCALL .ERASLN
TRAP CODE: $0027

Use .ERASLN to erase the line at the present cur

Entry Conditions:

No arguments required.

Exit Conditions:

The cursor is positioned at the beg

EXAMPLE

```
SYSCALL     .ERASLN
```


5.2.3 Check for Break

SYSCALL .CHKBRK
TRAP CODE: $0005

Returns zero (0) status in condition code registe
port.

Entry Conditions:

No arguments or stack allocation

Exit Conditions:

Z flag set in CCR if break detecte

EXAMPLE

```
SYSCALL     .CHKBRK
BEQ         BREAK
```

5.2.4 Timer Delay Function

SYSCALL .DELAY
TRAP CODE: $0043

The .DELAY function generates timing delays b the MCU periodic interrupt timer for operatic (number of interrupt pulses generated). .DELA specified delay is completed. Initialize (.TM_IN the .TM_RD function.

Entry Conditions:

SP ==> Delay time (numbe

Exit Conditions Different From Entry:

SP ==> The timer keeps ru
removed from the s

EXAMPLE

```
SYSCALL    .TM INI      Initialize
SYSCALL    .TM_STR0     Start time
PEA.L      &1500        Load a 1:
SYSCALL    .DELAY
*
*
*
PEA.L      &50000       Load a 5(
SYSCALL    .DELAY
```


5.2.5 Unsigned 32 x 32 Bit Divide

SYSCALL .DIVU32
TRAP CODE: $006A

Divide two 32-bit unsigned integers and return integer. The case of division by zero is handl $FFFFFFFF.

Entry Conditions:

SP ==> 32-bit divisor (valu
32-bit dividend (va
32-bit space for res

Exit Conditions:

SP ==> 32-bit quotient (res

EXAMPLE

Divide D0 by D1, load result into D2.

```
SUBQ.L     #4,A7        Allocate
MOVE.L     D0,-(A7)     Push divi
MOVE.L     D1,-(A7)     Push divi
SYSCALL    .DIVU32      Divide D
MOVE.L     (A7)+,D2     Get quoti
```

5.2.10 Unsigned 32 x 32 Bit Multiply

SYSCALL .MULU32
TRAP CODE: $0069

Multiply two 32-bit unsigned integers and retur
integer. No overflow checking is performed.

Entry Conditions:

SP ==> 32-bit multiplier
32-bit multiplicanc
32-bit space for res

Exit Conditions:

SP ==> 32-bit product (res

EXAMPLE

Multiply D0 by D1, load result into D2.

```
SUBQ.L     #4,A7        Allocate
MOVE.L     D0,-(A7)     Push mul
MOVE.L     D1,-(A7)     Push mul
SYSCALL    .MULU32      Multiply
MOVE.L     (A7)+,D2     Get prod
```


5.2.7 Input Character Routine

SYSCALL .INCHR
TRAP CODE: $0000

Reads a character from the default input port. Th

Entry Conditions:

SP ==> Space for characte
Word fill <byte>

Exit Conditions:

SP ==> Character <byte>
Word fill <byte>

EXAMPLE

```
SUBQ.L     #2,A7        Allo
SYSCALL    .INCHR       Call
MOVE.B     (A7)+,D0     Loa
```

.INLN Input Line

5.2.8 Input Line Routine

SYSCALL .INLN
TRAP CODE: $0002

Reads a line from the default input port. The min

Entry Conditions:

SP ==> Address of string b

Exit Conditions:

SP ==> Address of last cha

EXAMPLE

If A0 contains the string destination address:

```
SUBQ.L      #4,A7         Allocate
PEA         (A0)          Push poi
TRAP        #15           (May als
DC.W        2             macro ('’
MOVE.L      (A7)+,A1      Retrieve
```

NO

A line is a string of characters (<**CR**>). The maximum allowe terminating <**CR**> is not incluc Input/Output Control character pı 1.

.INSTAT Input Serial

5.2.9 Input Serial Port Status

SYSCALL .INSTAT
TRAP CODE: $0001

Checks the default input port buffer for charact result of the operation.

Entry Conditions:

No arguments or stack allocation

Exit Conditions:

Z (zero) = 1 if the receiver buffer

EXAMPLE

```
LOOP     SYSCALL     .INSTA
         BEQ.S       EMPTY
         SUBQ.L      #2,A7
         SYSCALL     .INCHR
         MOVE.B      (A7)+,
         BRA.S       LOOP
EMPTY
```

5.2.14 Read Line to Fixed-Length Buffer

SYSCALL .READLN
TRAP CODE: $0004

Reads a string of characters from the default ir
port. A string consists of a count byte followed l
byte indicates the number of characters read fron
the input string, excluding carriage return <**CR**>
254 characters.

Entry Conditions:

SP ==> Address of input bu

Exit Conditions:

SP ==> Top of stack
The first byte in th

EXAMPLE

If A0 points to a 256 byte buffer;

```
PEA       (A0)        Long buf
SYSCALL   .READLN     And read
```

NO

The caller must allocate 256 bytes
254 characters. <**CR**> and <**L**
following echo of the input. Se
character processing as described

5.2.11 Output Character Routine

SYSCALL .OUTCHR
TRAP CODE: $0020

Outputs a character to the default output port.

Entry Conditions:

SP ==> Character <byte>
Word fill <byte> (

Exit Conditions:

SP ==> Top of stack
Character is sent to

EXAMPLE

```
MOVE.B    D0,-(A7)    Send cha
SYSCALL   .OUTCHR     To defau
```

.OUTLN .OUTSTR

Output String U

5.2.12 Output String Using Pointers

SYSCALL .OUTLN
TRAP CODE: $0022

SYSCALL .OUTSTR
TRAP CODE: $0021

.OUTSTR outputs a string of characters to the c
characters followed by a <**CR**><**LF**> sequence.

Entry Conditions:

SP ==> Address of first ch
+4 Address of las

Exit Conditions:

SP ==> Top of stack

EXAMPLE

If A0 = start of string and A1 = end of string+1

```
MOVEM.L     A0/A1,-(A7)       Load poi
SYSCALL     .OUTSTR
```


.PCRLF

Print Carriage Ret

5.2.13 Print Carriage Return and Line Feed

SYSCALL .PCRLF
TRAP CODE: $0026

.PCRLF sends a carriage return and a line feed to

Entry Conditions:

No arguments or stack allocation

Exit Conditions:

None

EXAMPLE

```
SYSCALL     .PCRLF          Output
```

5.2.18 Compare Two Strings

SYSCALL .STRCMP

TRAP CODE: $0068

An equality comparison is made and a boolean f
identical the flag is $00, otherwise it is $FF.

Entry Conditions:

SP ==> Address of string#
Address of string#
Three bytes (unuse
Byte to receive stri

Exit Conditions:

SP ==> Three bytes (unuse
Byte that received

EXAMPLE

If A1 and A2 contain the addresses of the two str

```
SUBQ.L      #4,A7         Allocate
PEA         (A1)          Push add
PEA         (A2)          Push add
SYSCALL     .STRCMP       Compare
MOVE.L      (A7)+,D0      Pop bool
TST.B       D0            Check bc
BNE         ARE SAME      Branch i
```


5.2.15 Read String Into Variable-Length Buf

SYSCALL .READSTR

TRAP CODE: $0003

Reads a string of characters from the default inp
defines the maximum number of characters tha
should be no less than the first byte + 2. The ma
is 254 characters, making the maximum buffe
number of characters in the buffer. Enter a ca
terminate the input. The characters echo to the de

Entry Conditions:

SP ==> Address of input b

Exit Conditions:

SP ==> Top of stack
The count byte con

EXAMPLE

If A0 contains the string buffer address;

```
PEA         (A0)          Push buf
TRAP        #15           (May als
DC.W        3             macro (''
```

NO

This routine allows the caller to
input length (254 characters). I
entered, then the buffer inpu
Input/Output Control character pr
1.

.RETURN Return to (

5.2.16 Return to CPU32Bug

SYSCALL .RETURN
TRAP CODE: $0063

.RETURN restores control to CPU32Bug frc inserted in target code are removed. Then the Finally, the routine returns to CPU32Bug.

Entry Conditions:

No arguments required.

Exit Conditions:

Control is returned to CPU32Bug

EXAMPLE

```
SYSCALL     .RETURN           Return tc
```


.SNDBRK Send

5.2.17 Send Break

SYSCALL .SNDBRK
TRAP CODE: $0029

Use .SNDBRK to send a break to the default out

Entry Conditions:

No arguments or stack allocation

Exit Conditions:

The default port is sent ''break''.

EXAMPLE

```
SYSCALL     .SNDBRK
```

```
MOVE.L     #$00000002,-(A7)     F
SYSCALL    .TM_STR0             v
                                u
```

```
MOVE.L     #$054400A0,-(A7)     F
SYSCALL    .TM_STR0             (
                                v
```


5.2.19 Timer Initialization

SYSCALL .TM_INI
TRAP CODE: $0040

Use .TM_INI to initialize the MCU periodic inte
initializes it. .TM_INI does not restart the timer;
accomplished by counting the number of interrup
frequency is 125 milliseconds. Use this routine t

Entry Conditions:

No arguments required.

Exit Conditions Different From Entry:

Periodic interrupt timer is stopped
operation.

EXAMPLE

```
SYSCALL    .TM_INI          Initialize
```

5.2.20 Read Timer

SYSCALL .TM_RD
TRAP CODE: $0042

Use this routine to read the timer value (the
generated). Initialize (.TM_INI) and start (.TM
function.

Entry Conditions:

SP ==> Space for result <l

Exit Conditions Different From Entry:.

SP ==> Time (number of i
running after the re

EXAMPLE

```
SUBQ.L      #4,A7          Allocate
SYSCALL     .TM_RD         Read tim
MOVE.L      (A7)+,D0       Load inte
```


5.2.21 Start Timer at T=0

SYSCALL .TM_STR0
TRAP CODE: $0041

Use this routine to reset the timer to 0 and sta
periodic interrupt timer (periodic interrupt timin
register (PITR)), or use the default values. The
milliseconds and use level 6, vector 66. See App
Manual, MC68332UM/AD, concerning the Perio

Entry Conditions:

SP ==> Timer control valu
Timer period value

Exit Conditions Different From Entry:

Parameters are removed from the
counter is cleared. If the user's i
(SR), disables the timer interrupts
timer interrupts.

If the value of PICR is not equal
vector number is restored to the d

EXAMPLES

```
SYSCALL     .TM_STR0
```

```
MOVE.L      #0,-(A7)       Reset the
SYSCALL     .TM_STR0
```

. . . . . prints this message:

```
MOTOROLA QUALITY!
```

Using .WRITELN instead of .WRITE outputs thi

```
MOTOROLA
QUALITY!
```

NO

The string must be formatted su
pointed to by the passed address
string (pointer/count format – see

5.2.22 Output String with Data

SYSCALL	.WRITD	– Output
TRAP CODE:	$0028	
SYSCALL	.WRITDLN	– Output
TRAP CODE:	$0025	

These trap functions use the monitor I/O ro
embedded variable fields. .WRITD outputs a s
outputs a string of characters with data followe
passes the starting address of the string and the
inserted into the string. The output goes to the de

Entry Conditions:

Eight bytes of parameter positioned in the

SP ==> Address of string <
Data list pointer <l

A separate data stack or data list arranged

Data list pointer => Data for 1st variab
Data for next varia
Data for next varia

Exit Conditions:

SP ==> Top of stack (paran

EXAMPLE

The following section of code .....

```
ERRMESSG   DC.B           $15,'E
           MOVE.L         #3,-(A
           PEA            (A5)
           PEA            ERRMES
           SYSCALL        .WRITD
           TST.L          (A5)+
```

. . . . . prints this message:

```
ERROR CODE = 3
```

NO

The string must be formatted su pointed to by the passed address string, including the data field s see 5.1.2).

Format data fields withir '|<radix>,<fieldwidth>[Z]|' where base (in hexadecimal, i.e., "A" is l <fieldwidth> is the number of dat right-justified and left-most ch Include "Z" to suppress leading ze

All data is placed in the stack as l is encountered in the user string, data stack.

The data stack is not destroyed by (see example above) to de-alloca necessary for the space in the data be done using the call routine, as s

5.2.23 Output String Using Character Count

SYSCALL .WRITE – Outpu
TRAP CODE: $0023

SYSCALL .WRITELN – Outpu
TRAP CODE: $0024

.WRITE and .WRITELN format character string default output port. After formatting, the count b the starting address of the string. .WRITELN app

Entry Conditions:

Four bytes of parameters are posit

SP ==> Address of string.<

Exit Conditions:

SP ==> Top of stack (paran

EXAMPLE

```
MESSAGE1   DC.B           9,
MESSAGE2   DC.B           8,

           PEA            MESSAG
           SYSCALL        .WRITE
           PEA            MESSAG
           SYSCALL        .WRITE
```

6.2.13 Zero Pass Count (ZP)

Executing this command resets the pass counter entering a command that executes the loop-cont line as **LC** results in the pass counter being reset

6.3 UTILITIES

The monitor is supplemented by several utilities itself and the diagnostics.

6.3.1 Write Loop

WL.<SIZE> [<ADDR> [<DEL><DATA

The **WL** command executes a streamlined write This command is intended as a debugging aid o loop is very short in execution so measuring d tracking failures. Pressing the BREAK key doe ABORT switch or RESET switch does.

Command size must be specified as B for byte, V

The command requires two parameters: target a data are both hexadecimal values and must not l $10000, enter **WL.B 10000 00**. The system pr omitted.

<u>EXAMPLES</u>

CPU32Bug>**SD**<**CR**>

CPU32Diag>**WR.W**<**CR**>

CPU32Diag>**WR.B 40FC E6**<**CR**>

CPU32Diag>**WR.W 800C 43F6**<**CR**>

CPU32Diag>**WR.L 54F0 F8432191**<**CR**>

CHAP

DIAGNOSTIC FI

6.1 INTRODUCTION

This diagnostic guide contains operation inform Package, hereafter referred to as CPU32Diag. F user. Paragraphs 6.4 through 6.6 are guides to us

6.2 DIAGNOSTIC MONITOR

The tests described herein are called via a comm This monitor is command-line driven and prov error reporting, interrupt handling, and a multi-le

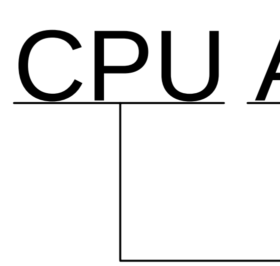
6.2.1 Monitor Start-Up

At the **CPU32Bug>** prompt, enter SD to sw Directories (SD) command is described elsewhe **CPU32Diag>**.

6.2.2 Command Entry and Directories

Enter commands at the **CPU32Diag>** prompt. Th carriage return <**CR**>. Multiple commands may and another command is to follow it, separat instance, to execute the MT B command after the MT A ! MT B. Spaces are not required but are sh be combined on one line.

Commands are listed in the diagnostic directory are listed in the directory for that particular comm



To execute a particular test, for example CPU,
This command causes the monitor to find the C
command from that subdirectory.

EXAMPLES

Single-Level Commands	HE
	DE
Two-Level Commands	CPU

6.2.3 Help (HE)

On-line documentation is provided in the form
name]). This command displays a menu of the to
a menu of each subdirectory if the name of that s
a menu of all the memory tests, enter **HE MT**.
pauses until the operator presses the carriage retu

6.2.4 Self Test (ST)

The monitor provides an automated test mecha
causes the monitor to run only the tests included
all the tests included in an internal self-test direc
parameters runs the entire directory, which conta

Each test for each particular command is listed in

6.2.5 Switch Directories (SD)

To exit the diagnostic directory (and disable the
diagnostic commands and initializes the CPU
directory, the prompt is **CPU32Bug>**. To ret
command. When in the diagnostic directory, the
user to access CPU32Bug without the diagnostic

6.2.6 Loop-On-Error Mode (LE)

Use the Loop-on-error mode (**LE**) to endlessl
detected. This is useful when using a logic anal
the test name to loop on errors encountered durin

6.2.7 Stop-On-Error Mode (SE)

Use the stop-on-error mode (**SE**) to halt a test a
then the test mnemonic to stop on errors encount

6.2.8 Loop-Continue Mode (LC)

Use loop-continue mode (**LC**) to endlessly repea
testing of everything on the command line. To t
diagnostic video display terminal. Certain tests d
ABORT or RESET switches of the M68300PFB

EXAMPLE

`CPU32Diag>`**`LC ST`**`<`**`CR`**`>`	Repeats se system.

6.2.9 Non-Verbose Mode (NV)

The diagnostics display a substantial number of
verbose mode (**NV**) suppresses all messages ex
NV, the test name, and <**CR**>. **NV ST MT** caus
only the names of the sub-tests and the results (p

6.2.10 Display Error Counters (DE)

Each test in the diagnostic monitor has a dedica
particular test, its error counter is incremented.
the test results could be determined by examinin
and a <**CR**> displays the results of a particular te

6.2.11 Clear (Zero) Error Counters (ZE)

The error counters, at start-up, initialize to a valu
to zero after errors have accumulated. The **ZE**
error counters can be individually reset by e
command. Example: **ZE CPU A** clears the error

6.2.12 Display Pass Count (DP)

A count of the number of passes in loop-contin
displayed with other information at the conclu
without using **LC**, enter **DP**.

CPU B Instructi

6.4.2 Instruction Test

CPU32Diag>**CPU B**

CPU B tests various data movement, integer
manipulation instructions of the MCU device.

EXAMPLE

After the command has been issued, the f

```
B     CPU Instruction Test ......
```

If any part of the test fails, then the displa

```
B     CPU Instruction Test.......
```

(error message)

Here, (error message) is one of the follow

```
Failed AND/OR/NOT/EOR instruct
Failed DBF instruction check
Failed ADD or SUB instruction
Failed MULU or DIVU instructio
Failed BSET or BCLR instructio
Failed LSR instruction check
Failed LSL instruction check
```

If all parts of the test are completed corre

```
B     CPU Instruction Test.......
```


6.3.2 Read Loop

RL.<SIZE> [<ADDR> [<DEL><DATA

The **RL** command executes a streamlined rea
location. This command is intended as a debugg
The read loop is very short in execution so m
utilized in tracking failures. Pressing the BREA
pressing the ABORT switch or RESET switch do

Command size must be specified as B for byte, W

The command requires one parameter: target ad
read from address $10000, enter **RL.B 10000**. T
omitted.

EXAMPLES

CPU32Diag>**RL.B**<**CR**> Pron

CPU32Diag>**RL.W A000**<**CR**> Rea

6.3.3 Write/Read Loop

WR.<SIZE> [<ADDR> [<DEL><DATA

The **WR** command executes a streamlined write
location. This command is intended as a debugg
The write/read loop is very short in execution so
utilized in tracking failures. Pressing the BREA
pressing the ABORT switch or RESET switch do

Command size must be specified as B for byte, W

The command requires two parameters: target a
data are both hexadecimal values and must not b
$10000 and read back, enter **WR.B 10000 00**.
parameters are omitted.

EXAMPLE

CPU32Diag>**WR.W 8000 FFFFFFFF**<**CR**> Wri
read

6.4 CPU TESTS FOR THE MCU

CPU tests are a series of diagnostics used to te below (Table 6-1).

Table 6-1. MCU CI

Monitor Command
CPU A
CPU B
CPU C
CPU D

The normal procedure for correcting a CPU erro

6.4.1 Register Test

CPU32Diag>**CPU A**

CPU A executes a thorough test of all the regis bits stuck high or low.

EXAMPLE

After the command has been issued, the f

```
A     CPU Register test..........
```

If any part of the test fails, then the displa

```
A     CPU Register test..........
```
(error message)

Here, (error message) is one of the follow

```
Failed DO-D7 register check
Failed SR register check
Failed USP/VBR/CAAR register c
Failed CACR register check
Failed AO-A4 register check
Failed A5-A7 register check
```

If all parts of the test are completed corre

```
A     CPU Register test..........
```

The following describes the memory error displa
reporting code is designed to conform to two rul

1. The first time an error occurs, headi values.
2. Upon 20 memory errors, the printing test.

The memory error display format is:

```
FC     TEST ADDR   10987654321098765432 1
5      00010000    ---------------------
5      00010004    -------------------X-
```

Each line displayed consists of five items: f expected data, and read data. The test address, hexadecimal. The graphic bit report shows a lett each good bit position.

The heading used for the graphic bit report is int Each numeral in the heading is the one's digit o bit at test address $10004 has the numeral 2 ove the bit position is read 12 in decimal (base 10).

CPU C

6.4.3 Address Mode Test

CPU32Diag>**CPU C**

CPU C tests the various addressing modes of th address indirect, address indirect with post-incre

EXAMPLE

After the command has been issued, the f

```
C     CPU Address Mode test......
```

If any part of the test fails, then the displa

```
C     CPU Address Mode test......
```

(error message)

(error message) is one of the following:

```
Failed Absolute Addressing che
Failed Indirect Addressing che
Failed Post increment check
Failed Pre decrement check
Failed Indirect Addressing wit
Unexpected Bus Error at $XXXXX
```

If all parts of the test are completed corre

```
C     CPU Address Mode test......
```

6.4.4 Exception Processing Test

CPU32Diag>**CPU D**

CPU D tests many of the exception processing
vectors or any of the floating point co-processor

EXAMPLE

After the command has been issued, the f

```
D     CPU Exception Processing Te
```

If any part of the test fails, then the displa

```
D     CPU Exception Processing Te
Test Failed Vector # XXX
```

XXX is the hexadecimal exception vec
Manual.

However, if the failure involves taking a
display is:

```
D     CPU Exception Processing Te
Unexpected exception taken to Vec
```

If all parts of the test are completed corre

```
D     CPU Exception Processing Te
```


6.5 MEMORY TESTS (MT)

The memory tests are a series of diagnostics w
that may or may not reside on the M68300EVS
RAM. To test off-board RAM, change Start a
described in the following paragraphs. Memory t

NO

If one or more memory tests are atter
memory, a bus error message appears, giv

Table 6-2. Memor

MONITOR COMMAND
MT A
MT B
MT C
MT D
MT E
MT F
MT G
MT H
MT I
MT J

The following hardware is required to perform th

- M68300EVK - Module being tested
- Video display terminal or host compu

6.5.4 Set Bus Data Width

CPU32Diag>**MT D** [new value: 0 for 16,

MT D selects either 16-bit or 32-bit bus data ac
tests. The width is selected by entering zero for 1

EXAMPLE

If the user supplied the optional new valu

```
CPU32Diag>MT D [new value]<CR>
Bus Width (32=1/16=0) =<new value
CPU32Diag>
```

If a new value was not specified by the u
is allowed to enter a new value.

NO

The default value is Bus

```
CPU32Diag>MT D<CR>
Bus Width (32=1/16=0) =<current v
Bus Width (32=1/16=0) =<new value
CPU32Diag>
```

This command may be used to display th
carriage return <CR> without entering th

```
CPU32Diag>MT D<CR>
Bus Width (32=1/16=0) =<current v
Bus Width (32=1/16=0) =<current v
CPU32Diag>
```


6.5.1 Set Function Code

CPU32Diag>**MT A** [new value]

MT A allows the user to select the function code
this are Program Test and **TAS** Test.

EXAMPLE

If the user supplied the optional new valu

```
CPU32Diag>MT A [new value]<CR>
Function Code=<new value>
CPU32Diag>
```

If a new value was not specified by the
user is allowed to enter a new value.

NO

The default is Function Code=

```
CPU32Diag>MT A<CR>
Function Code=<current value> ?[n
Function Code=<new value>
CPU32Diag>
```

This command may be used to display th
carriage return <CR> without entering th

```
CPU32Diag>MT A<CR>
Function Code=<current value> ?<C
Function Code=<current value>
CPU32Diag>
```

6.5.2 Set Start Address

CPU32Diag>**MT B** [new value]

MT B allows the user to select the start add
MVME332, it is suggested that address $00003
extreme caution should be used when attempting

EXAMPLE

If the user supplied the optional new valu

```
CPU32Diag>MT B [new value]<CR>
Start Addr.=<new value>
CPU32Diag>
```

If a new value was not specified by the u
is allowed to enter a new value.

NO

The default is Start Addr.=00003

```
CPU32Diag>MT B<CR>
Start Addr.=<current value> ?[new
Start Addr.=<new value>
CPU32Diag>
```

This command may be used to display th
carriage return <CR> without entering th

```
CPU32Diag>MT B<CR>
Start Addr.=<current value> ?<CR>
Start Addr.=<current value>
CPU32Diag>
```

NO

If a new value is specified, it is t
and, if greater than the value of t
address. The start address is neve
the stop address. These changes c
processed by the monitor.

6.5.3 Set Stop Address

CPU32Diag>**MT C** [new value]

MT C allows the user to select the stop address u
is the address where testing terminates, so the sto

EXAMPLE

If the user supplied the optional new valu

```
CPU32Diag>MT C [new value]<CR>
Stop Addr.=<new value>
CPU32Diag>
```

If a new value was not specified by the u
is allowed to enter a new value.

NO

The default is Stop Addr.=00010
RAM.

```
CPU32Diag>MT C
Stop Addr.=<current value> ?[new
Stop Addr.=<new value>
CPU32Diag>
```

This command may be used to display th
carriage return <CR> without entering th

```
CPU32Diag>MT C
Start Addr.=<current value> ?<CR>
Start Addr.=<current value>
CPU32Diag>
```

NO

If a new value is specified, it is t
and, if less than the value of the
start address. The stop address
memory than the start address. Th
command is processed by the mon

6.5.8 Random Byte Test

CPU32Diag>**MT H**

MT H performs a random byte test from Start
has been implemented in this manner:

1. A register is loaded with the value $E
2. For each memory location:
   - Copy the contents of the re
   - Add $02468ACE to the co
   - Proceed to next memory lo
3. Reload $ECA86420 into the register.
4. For each memory location:
   - Compare the contents of
     contents are good, one byte
   - Add $02468ACE to the co
   - Proceed to next memory lo

EXAMPLE

After the command is entered, the display

```
H     MT Random Byte Test........
```

If an error occurs, then the memory locati

```
H     MT Random Byte Test........
```

(error-related information)

If no errors occur, then the display appear

```
H     MT Random Byte Test........
```


6.5.5 March Address Test

CPU32Diag>**MT E**

MT E performs a march address test from Start A
has been implemented in this manner:

1. All memory locations from Start Add
2. Beginning at Stop Address and proce
   location is checked for bits that did n
   F's (all the bits are set). This process
3. Beginning at Start Address and proc
   location is checked for bits that did
   cleared to 0. This process reveals add

EXAMPLE

After the command is entered, the display

```
E     MT March Addr. Test........
```

If an error is encountered, then the mem
displayed.

```
E     MT March Addr. Test........
```

(error-related information)

If no errors are encountered, then the disp

```
E     MT March Addr. Test........
```

6.5.6 Walk a Bit Test

CPU32Diag>**MT F**

MT F performs a walking bit test from start addr
memory location is implemented in the following

- Write out a 32-bit value with only the
- Read it back and verify that the value
- Shift the 32-bit value to move the bit
- Repeat the procedure (write, read, and

EXAMPLE

After the command is entered, the display

```
F     MT Walk a bit Test ........
```

If an error is encountered, then the mem
displayed.

```
F     MT Walk a bit Test ........
```
(error-related information)

If no errors are encountered, then the disp

```
F     MT Walk a bit Test ........
```


6.5.7 Refresh Test

CPU32Diag>**MT G**

MT G performs a refresh test from Start Addr
implemented in this manner:

1. For each memory location:
   - Write out value $FC84B73
   - Verify that the location con
   - Proceed to next memory lo
2. Delay for 500 milliseconds (1/2 secon
3. For each memory location:
   - Verify that the location con
   - Write out the complement o
   - Verify that the location con
   - Proceed to next memory lo
4. Delay for 500 milliseconds.
5. For each memory location:
   - Verify that the location con
   - Write out value $FC84B73
   - Verify that the location con
   - Proceed to next memory lo

EXAMPLE

After the command is entered the display

```
G     MT Refresh Test...........
```

If an error is encountered, then the mem
displayed.

```
G     MT Refresh Test...........
```
(error-related information)

If no errors are encountered, then the disp

```
G     MT Refresh Test...........
```

6.5.9 Program Test

CPU32Diag>**MT I**

MT I moves a program segment into RAM and e

1. The program is moved into the RAM
the available RAM (i.e., from Star
segments of the program are moved
segment copied into the RAM to S
Attempting to run this test without su
one complete program segment to b
out: INSUFFICIENT MEMORY.
2. The last location, Stop Address, recei
3. Finally, the test performs a JSR to loc
4. The program itself performs a wide v
checked and a count of the errors i
fashion as any memory test failure (re

EXAMPLE

After the command is entered, the display

```
I     MT Program Test...........
```

If the operator has not allowed enough r
copied into the target RAM, then the fol
make sure that the Stop Address is at lea
Address.

```
I     MT Program Test...........
      Insufficient Memory
      PASSED
```

If the program (in RAM) detects any e
information is displayed.

```
I     MT Program Test...........
```
(error-related information)

If no errors occur, then the display appear

```
I     MT Program Test...........
```

6.5.10 Test and Set Test

CPU32Diag>**MT J**

MT J performs a Test and Set (TAS) test from
memory location is implemented as follows:

- Clear the memory location to 0.
- Test And Set the location (should set
- Verify that the location now contains
- Proceed to next location (next byte).

EXAMPLE

After the command is entered, the display

```
J     MT TAS Test.................
```

If an error occurs, then the memory locati

```
J     MT TAS Test.................
```
(error-related information)

If no errors occur, then the display appear

```
J     MT TAS Test.................
```


6.6 BUS ERROR TEST

CPU32Diag>**BERR**

BERR tests for local bus time-out and global b
following:

- No bus error by reading from ROM
- Local bus time-out by reading from a
- Local bus time-out by writing to an u

EXAMPLE

After the command has been issued, the f

```
BERR  Bus Error Test.............
```

If a bus error occurs in the first part of th
as follows.

```
BERR  Bus Error Test.............
Got Bus Error when reading from R
```

If no bus error occurs in one of the oth
appropriate error message appears as one

```
   No Bus Error when reading from
   No Bus Error when writing to B
```

If all three parts of the test are completed

```
BERR  Bus Error Test.............
```

The next 16 character pairs of the first S1 rec
code/data. In this assembly language example,
written in sequence in the code/data fields of the

OPCODE	INSTRUC
285F	MOVE.L
245F	MOVE.L
2212	MOVE.L
226A0004	MOVE.L
24290008	MOVE.L
237C	MOVE.L

(The balance of this code is conti
records and stored in memory.)

2A The checksum of the first S1 reco

The second and third S1 records also each cont
checksums 13 and 52 respectively. The fourth
checksum of 92.

The S9 record is explained as follows:

S9	S-record type S9, indicating that it is a te
03	Hexadecimal 03, indicating that three cha
00 00	The address field, zeros.
FC	The checksum of the S9 record.

Each printable character in an S-record is enc
representation of the binary bits which are actua
above is sent as:

TYPE				LENGTH				ADDRESS						
S		1		1		3		0		0		0		
5	3	3	1	3	1	3	3	3	0	3	0	3	0	
0101	0011	0011	0001	0011	0001	0011	0011	0011	0000	0011	0000	0011	0000	0

APPEN

S-RECORD IN

A.1 INTRODUCTION

The S-record format for output modules was de
data files in a printable format for transportation
process can thus be visually monitored and the S

A.2 S-RECORD CONTENT

When viewed by the user, S-records are essential
identify the record type, record length, memory
binary data is encoded as a 2-character hexadeci
high-order 4 bits, and the second the low-order 4

The five fields which comprise an S-record are s

TYPE	RECORD LENGTH	ADDRE

Where the fields are composed as follows:

Field	Printable Characters	
type	2	S-records type -- S0
record length	2	The count of the cha length.
address	4, 6, or 8	The 2-, 3-, or 4-byte memory.
code/data	0-n	From 0 to n bytes of descriptive informati programs may limit t characters in the S-r
checksum	2	The least significant values represented b length, address, and

Each record may be terminated with a CR/LF/ initial field to accommodate other data such as systems. An S-record file is a normal ASCII text

Accuracy of transmission is ensured by the recor

A.3 S-RECORD TYPES

Eight types of S-records have been defined to a transportation and decoding functions. The vario transportation control programs, as well as cros debugging programs, utilize only those S-record specific information on which S-records are supp for the program must be consulted. CPU32Bug s

An S-record format module may contain S-recor

S0	The header record for each block of S-recor information identifying the following block of
S1	A record containing code/data and the 2-byt
S2	A record containing code/data and the 3-byt
S3	A record containing code/data and the 4-byt
S5	A record containing the number of S1, S2, a count appears in the address field. There is
S7	A termination record for a block of S3 record byte address of the instruction to which cont
S8	A termination record for a block of S2 record byte address of the instruction to which cont
S9	A termination record for a block of S1 record byte address of the instruction to which cont specification encountered in the object mod

Only one termination record is used for each bl used only when control is to be passed to a 3 or 4 is used, although it is possible for multiple heade

A.4 S-RECORDS CREATION

S-record format files may be produced by du assemblers or cross linkers. Several programs a format from a host system to a microprocessor-b

EXAMPLE

Shown below is a typical S-record format modul

```
S00600004844521B
S1130000285F245F2212226A00042429
S11300100002000800082629001853812
S113002041E900084E42234300182342
S113003000144ED492
S9030000FC
```

The module consists of one S0 record, four S1 re

The S0 record is comprised of the following char

S0	S-record type S0, indicating that it is a he
06	Hexadecimal 06 (decimal 6), indicating th
00 00	Four-character, 2-byte, address field; zero
48 44 52	ASCII H, D and R - "HDR".
1B	The checksum.

The first S1 record is explained as follows:

S1	S-record type S1, indicating that it is a co address.
13	Hexadecimal 13 (decimal 19), indicating of binary data, follow.
00 00	Four-character, 2-byte, address field; hex follows is to be loaded.

C.2 CPU32BUG CUSTOMIZATION

The general procedure for customizing CPU32B

1. Copy the parameter area from the following command:

```
CPU32Bug>BM E0000 E01
```

2. Modify the parameters in RAM usir the CHECKSUM value would begi Thus the word at $400E must be cha customized CPU32Bug can be calcu the CHECKSUM value.

```
CPU32Bug>MS 400E FFFF
```

   Change the SIGNON message to ind the spaces after "Version 1.01" customized version number starting or school/lab. Use the **MS** command

3. Create an S-record file of the change key on the host computer terminal (file. Enter the file name **C32B1.** command by pressing <**CR**>. The of with the proper starting address of $I

```
CPU32Bug>DU 4000 41FF
CPU32Bug><ALT-F1><CR>
```

4. Create an S-record file of the rest of ALT-F1 key on the host computer open a log file. Enter the file name **DU** command by pressing <**CR**>.

```
CPU32Bug>DU E0200 FFF
CPU32Bug><ALT-F1><CR>
```

5. If desired, the two S-record files ca "Effective address" lines at the begi the end, but it is not required. If the t edit the first file to remove the S8 te

APPEN

SELF-TEST ERF

B.1 INTRODUCTION

On power-up or reset, CPU32Bug executes a sy integrity before issuing the sign-on message (SIG error is detected, testing is aborted and an error r monitor prompt. Error messages are summarized "000EXXXXXX" because the actual error a personnel. Additional error values, such as addre

Table B-1. Self-Te

Test Type and Error Messa
CPU Register Test:
ERROR $01 @ $000EXXXX, CONFIDENCE
ERROR $02 @ $000EXXXX, CONFIDENCE
ERROR $03 @ $000EXXXX, CONFIDENCE
ERROR $04 @ $000EXXXX, CONFIDENCE
ERROR $05 @ $000EXXXX, CONFIDENCE
ERROR $06 @ $000EXXXX, CONFIDENCE
ERROR $07 @ $000EXXXX, CONFIDENCE
CPU Instruction Test:
ERROR $10 @ $000EXXXX, CONFIDENCE
ERROR $11 @ $000EXXXX, CONFIDENCE
ERROR $12 @ $000EXXXX, CONFIDENCE
ERROR $13 @ $000EXXXX, CONFIDENCE
ERROR $14 @ $000EXXXX, CONFIDENCE
ERROR $15 @ $000EXXXX, CONFIDENCE
ERROR $16 @ $000EXXXX, CONFIDENCE

Table B-1. Self-Test Err

Test Type and Error Messa
ROM Test:
ERROR $20 @ $000EXXXX, CONFIDENCE
ERROR $21 @ $000EXXXX, CONFIDENCE
RAM Test:
ERROR $30 @ $000EXXXX, CONFIDENCE
CPU Addressing Test:
ERROR $40 @ $000EXXXX, CONFIDENCE
ERROR $41 @ $000EXXXX, CONFIDENCE
ERROR $42 @ $000EXXXX, CONFIDENCE
ERROR $43 @ $000EXXXX, CONFIDENCE

APPEN

USER CUST

C.1 INTRODUCTION

Within the CPU32Bug certain operating parame
situation. This appendix details the customiza
compatible host computer with the Motorola
required to reprogram the EPROM on the BC
ProComm terminal emulation program on the h
and is familiar with the following; CPU32Bug, P

NO

In the back of this appendix is a
may be helpful to refer to the Q
CPU32Bug.

CAU

Failure to incorporate changes a
cause malfunctions in the CPU32
to customize CPU32Bug.

The user customization area (parameter area)
$E01FF), see Table C-1. For brevity's sake, all e
the $E0000 base address of CPU32Bug. The so
initialization table, and chip select initializa
FREEWARE Bulletin Board Service (BBS) un
updates for CPU32Bug will also be available
filename C32xxx.ARC. For more information
letter M68332EVS/L2.

Because there are two versions of the M68332B
set for Rev. A and one set for Rev. B. Upon
CSBOOT chip select and CS0/CS1 (see Rev. A
determine if the hardware is Rev. A or Rev. B. C
values from the proper table. The only changes r
BASE ADDRESS fields for their platform board

Table C-1. CPU32Bug Cust

Offset	Default Value	Mnemonic	
Common Chip Select Table: (Rev. B BCC -			
$3C-3D	$0E04	.CSBARBT	CSE
$3E-3F	$68B0	.CSORBT	.
New Chip Select Table: (F			
$40-41	$0003	.CSBAR0	CS0
$42-43	$503E	.CSOR0	.
$44-45	$0003	.CSBAR1	CS1
$46-47	$303E	.CSOR1	.
$48-49	$0003	.CSBAR2	CS2
$4A-4B	$683E	.CSOR2	.
$4C-4D	$0000	.CSBAR3	CS3
$4E-4F	$0000	.CSOR3	.
$50-51	$FFF8	.CSBAR4	CS4
$52-53	$680F	.CSOR4	.
$54-55	$FFE8	.CSBAR5	CS5
$56-57	$783F	.CSOR5	.
$58-59	$1004	.CSBAR6	CS6
$5A-5B	$38F0	.CSOR6	.
$5C-5D	$1004	.CSBAR7	CS7
$5E-5F	$58F0	.CSOR7	.
$60-61	$0103	.CSBAR8	CS8
$62-63	$6870	.CSOR8	.
$64-65	$0103	.CSBAR9	CS9
$66-67	$3030	.CSOR9	.
$68-69	$0103	.CSBAR10	CS1
$6A-6B	$5030	.CSOR10	.
$6C-6D	$020F	MCR_OR	Valu on/r
$6E-6F	$DFFF	MCR_AND	Valu stor 0, th Oth

6. Verify the customized S-record file below. The -DC000 offset is requir base address of the S-records to the I

   `CPU32Bug>`**`VE -DC000`**`<`**`CR`**`>`

   Enter the terminal emulator's escape system (ALT-F4 for ProComm). The file to the port where the BCC is con connected to the com1 port).

   After the file has been sent, restart th on the host computer. Then ente verification is complete and the ter status message.

   ```
   <CR><CR>
   Verify passes.
   CPU32Bug>
   ```

7. Verify the main S-record file, **C32B** No offset is required.

   `CPU32Bug>`**`VE`**`<`**`CR`**`>`

   Enter the terminal emulator's escape system (ALT-F4 for ProComm). The S-record file to the BCC (**type c32b** com1 port).

   After the file has been sent, restart t host computer. Then enter two <**CR** complete and the terminal emulator p

   ```
   <CR><CR>
   Verify passes.
   CPU32Bug>
   ```

8. Follow the PROGBCC utility reprogramming the BCC EPROM **C32B23.MX**.

9. Power up the newly programmed Repeat steps 1 through 8 above, t changes noted below. The CODESIZ the checksum valid only over the CP second half can be freely changed. S display terminal and code executi checksum.

STEP 1: No change.

STEP 2: Change checksum to th below where "XXXX" is

```
CPU32Bug>MS 400E XXX
```

STEP 3: Change the filename to temporary file consistin entering **DU 400E 400** creating the C32B1C.MX

STEP 4: Skip this step.

STEP 5: No change.

STEP 6: Change the filename to **C**

STEP 7: This step is optional.

STEP 8: Only the checksum valu value. Since the checks EPROM ($FFFF), pro ERASE THE BCC EPR

10. Power-up the BCC once again. The c

11. On the host computer, enter the follov record files so they may be properly a

```
C>DEL TMP.MX<CR>
C>DEL C32B1.MX<CR>
C>RENAME C32B1C.MX C32B1.MX
C>COPY C32B*.MX A:<CR>
```

12. The customization procedure is now c

C.3 CUSTOMIZATION TABLE

Table C-1. CPU32Bug

Offset	Default Value	Mnemonic	
$00-03	$00002FFC	PWR_SSP	Pow
$04-07	$000E0090	PWR_PC	Pow
$08-0B	$00020000	CODESIZE	Size N a
$0C	$20	SRECMAX	Max D L
$0D	$FF	CHECKALT	Che C c
$0E-0F	$3033	CHECKSUM	Che $ c s r
Old Chip Select Table (Re			
$10-11 $12-13	$0003 $5830	.CSBAR0 .CSOR0	CS0 .
$14-15 $16-17	$0003 $3830	.CSBAR1 .CSOR1	CS1 .
$18-19 $1A-1B	$0103 $6870	.CSBAR2 .CSOR2	CS2 .
$1C-1D $1E-1F	$0103 $3030	.CSBAR3 .CSOR3	CS3 .
$20-21 $22-23	$1004 $5870	.CSBAR4 .CSOR4	CS4 .
$24-25 $26-27	$1004 $3870	.CSBAR5 .CSOR5	CS5 .
$28-29 $2A-2B	$FFE8 $783F	.CSBAR6 .CSOR6	CS6 .
$2C-2D $2E-2F	$0000 $0000	.CSBAR7 .CSOR7	CS7 .
$30-31 $32-33	$FFF8 $680F	.CSBAR8 .CSOR8	CS8 .
$34-35 $36-37	$0000 $0000	.CSBAR9 .CSOR9	CS9 .
$38-39 $3A-3B	$0103 $5030	.CSBAR10 .CSOR10	CS1 .

Table C-1. CPU32Bug Cust

Offset	Default Value	Mnemonic	
Power On Branch V			
$90-95	$60FF0000E056	PWR_TBL1	BRA belo
$96-9B	$60FF0000DEE8	PWR_INI	BRA Ex Re
$9C-A1	$60FF0000E070	PWR_TBL2	BRA be Ex
$A2-A7	$60FF00000004	PWR_TTL	BRA Re Ex
$A8-AD	$60FF0000D8AA	PWR_TST	BRA Ex DT Re
$AE-B3	$60FF0000D4B4	PWR_GO	BRA En th DT Ne
$B4-B9	all $FF's		BRA
$BA-BF	all $FF's		BRA
$C0-CF	all $FF's		<res

Table C-1. CPU32Bug Cust

Offset	Default Value	Mnemonic	
$70	$06	SYPCR_OR	Valu up/r
$71	$FF	SYPCR_AND	Valu stor write halt

NO

Enabling the software watchdog with CPU32Bug itself to fail when the watch failure is constant RESETing before RESETing during execution of particula

Disabling the bus monitor timeout pe any unterminated bus cycle, i.e., accessi

Changing the bus monitor timeout pe problems with slow memory or if the 8-

Offset	Default Value	Mnemonic	
$72-73	$8000	FCRYSTAL	Crys rate
$74-77	$FFFFFFFF	FEXTAL	Exte MOD EXT valu

Table C-1. CPU32Bug Cust

Offset	Default Value	Mnemonic	
ROM AUTO BC			
$78-7B	$FFFFFFFF	RB_SP	ROI
$7C-7F	$FFFFFFFF	RB_PC	ROI Bit (
CONSOLE DEFAULT TA			
$80-83	$00001C0F	.PARMS	Para D
$84-85	$2580	.BAUD	Bau
$86	$00	.PARITY	Pari
$87	$08	.DATA	Data

Table C-1. CPU32Bug Cust

Offset	Default Value	Mnemonic	
Console Default Table for S			
$88	$01	.STOP	Stop
$89	$FF	.XON_ENB	XON
$8A	$11	.XON	XON
$8B	$13	.XOFF	XOF
Periodic Inte			
$8C-8D	$0642	.PICR	Peri D
$8E-8F	$0102	.PITR	Peri C fu m

C.4 COMMUNICATION FORMATS

Not all combinations of data bits, parity, and s
details the legal combinations that can be used w

Table C-2. MCU SCI C

Character Width	Parity
7	None
7	None
7	Even
7	Even
7	Odd
7	Odd
8	None
8	None
8	Even
8	Even
8	Odd
8	Odd

Table C-1. CPU32Bug Cust

Offset	Default Value	Mnemonic	
Initializati			
$D0-16F	all $FF’s	INITTBL	Initia

The Initialization Table is organized as
following format:

<ADDR> <CNT/SZ> <FILL> <
4 1 0|1

Where:

<ADDR> is the destination addres
start on a even address (
value ($FFFFFFFF) term

<CNT/SZ> is the count/size code for

n is the upper nibb
number of <DAT
successive addres

s is the lower nibb
and the storage o
1 = BYTE
2 = WOR
4 = LONG

An invalid size c

<FILL> is a dummy placeholde
LONG WORD sized <
address (word) boundar
data, otherwise it is one b

<DATA> is the byte, word, or lon
to be stored starting at
data bytes. If the data siz
of <DATA> elements (n
next Table entry will star

Table C-1. CPU32Bug Cust

Initialization Ta

This entry format aligns with the normal
automatically aligned on an an even address (w
Thus the <FILL> byte is handled automatically

	Rel. Addr	Contents	Label
	0000	00FFFFA21	
	0004	01	
	DATA		
	0005	04	
	0006	00FFFFA21	
	000A	31	
	DATA		
	000B	04 22 47 FE	
Skips $1F->	0010	00FFFFA22	
	0014	02	
	DATA		
Skips $15->	0016	0544	
	0018	00FFFFA74	
	001C	04	
	DATA		
Skips $1D->	001E	12345678	
	0022	00FFFFA74	
	0026	04	
	DATA		
Skips $27->	0028	12345678	
	002C	0002307F	
	0030	FFFFFFFF Terminate	

The routine will also terminate before any atte
end of the table. Thus the user can comple
termination entry whose <ADDR> equals FILI

Table C-1. CPU32Bug Cust

Offset	Default Value	Mnemonic	
Sign On Te			
$170-1FF		SIGNON	Text

Default values shown in MASM assembly l
substituted for each space character (" ") to sho
be preserved.

```
SIGNON    DC.B     SIGN$2-SIGN$1
SIGN$1    DC.B     $0D,$0A,$0A
          DC.B     'CPU32Bug^Debugg
          DCB.B    34,$20
          DC.B     $0D,$0A
          DC.B     '^(C)^Copyright,^19
          DCB.B    23,$20
SIGN$2    EQU      *
```

C.8 PLATFORM BOARD (PFB) REV

PFB Rev. C boards have jumpers (J8 - J13 compatible with Rev. A, Rev. B or Rev. C BCC to Rev. B or C compatibility on a Rev. C PFB, al

Table C-6. PFB Re

BCC BOARD REVISION	PFB Rev. A	PFB Rev. B
BCC Rev. A	YES	NO
BCC Rev. B	NO	YES
BCC Rev. C	NO	YES
(1) The default when no jumper block is installed is F		

C.5 BCC REV. A CHIP SELECTION

Table C-3 covers Rev. A of the M68332BCC Bu Platform Board.

Table C-3. Rev. A Chi

Signal	Board/Chip	
CSBOOT	BCC U4	CPU32Bug EPROM
CS0	BCC U3	read/write enable for
CS1	BCC U2	read/write enable for
CS2	PFB U1/U3	read enable for MSB
CS3	PFB U1	write enable for LSB
CS4	PFB U4	read enable for MSB
CS5	PFB U2	read enable for LSB
CS6	PFB U5	chip enable for MC6
CS7	<unused>	
CS8	PFB	ABORT pushbutton
CS9	<unused>	
CS10	PFB U3	write enable for MSB cut/jump U3-27 from

NO

U1/U3 = 120 nsec RAM with fast termi

U2/U4 = ROM laid-out wrong, can only

C.6 BCC REV. B CHIP SELECTION

Table C-4 covers Rev. B of the M68332BC
Platform Board.

Table C-4. Rev. B Chi

Signal	Board/Chip	
CSBOOT	BCC U4	CPU32Bug EPROM
CS0	BCC U3	write enable for MSE
CS1	BCC U2	write enable for LSB
CS2	BCC U2/U3	read enable for MSE
CS3	<unused>	
CS4	PFB	ABORT pushbutton
CS5	PFB U5	chip enable for MC6 CS2 to CS5 requirec
CS6	PFB U2	read enable for LSB
CS7	PFB U4	read enable for MSE
CS8	PFB U1/U3	read enable for MSE
CS9	PFB U1	write enable for LSB
CS10	PFB U3	write enable for MSE

NO

U1/U3 = 120 nsec RAM with fast termi

U2/U4 = 250 nsec EPROM (or jumper s

C.7 BCC REV. C CHIP SELECTION

The table below covers Rev. C of the M68332F
Platform Board.

Table C-5. BCC Rev. C C

Signal	Board/Chip	
CSBOOT	BCC U3	CPU32Bug EPROM
CSBOOT	BCC U4	CPU32Bug EPROM
CS0	BCC U1	write enable for MSE
CS1	BCC U2	write enable for LSB
CS2	BCC U3/U1	read enable for MSE
CS3	<unused>	
CS4	PFB	ABORT push-button
CS5	PFB U5	chip enable for MC6 CS2 to CS5 requirec
CS6	PFB U2	read enable for LSB
CS7	PFB U4	read enable for MSE
CS8	PFB U1/U3	read enable for MSE
CS9	PFB U1	write enable for LSB
CS10	PFB U3	write enable for MSE

Q: How can I get CPU32Bug to automatically e

A: Use the ROM Auto Boot Vectors (**RB_SP** ar
whereby CPU32Bug initializes itself and the
counter (PC), thus starting execution of the u

C.9 CPU32BUG QUESTIONS AND A

Q: How can I change the chip selections to fit m

A: Use the Chip Select Table parameters to cust
two tables; an Old one for Rev. A BCC units
selection is based upon whether good RAM i
programmed using the Old Table values. Co
assignments. The chip selects designated for
selects can be used if the corresponding reso
correct table, or place them in both tables if y

Q: How can I change CPU32Bug to automatica
Standby RAM Module on the MC68332, upo

A: Use the Initialization Table (INITTBL) to se
will initialize the desired resource. The follo
can be used to initialize the 2K Standby RAM
$80000 in unrestricted space and assumes th
(MM bit in MCR register equals one). Reme
the normal chip select initialization (via **PW**
after the normal chip select initialization.

Offset	Value	
$D0	$FFFFFFFF	Table
$D4	$00FFFB00	RAM
$D8	$02	Word
$D9	$FF	Filler
$DA	$0000	Word
$DC	$00FFFB04	RAM
$E0	$02	Word
$E1	$FF	Filler
$E2	$0800	Word
$E4	$FFFFFFFF	Table

Q: How can I change CPU32Bug so I don't hav
time I change my user program in the second

A: Change the **CODESIZE** parameter to $1000
used in calculating the checksum. Or, disable
unprogrammed state of all $FF's, i.e., set the

Q: How can I change the Periodic Interrupt Tim
functions?

A: Change the Periodic Interrupt Timer **.PITR** p
parameter's value is placed into the PITR reg

Q: How can I change the default RS-232 commu
number of data bits, stop bits, and XON/XOF

A: Use the Console Default Table for SCI (**.BAU**
.XON, and **.XOFF**) to change these paramete

Q: How can I change the crystal frequency? Ca

A: Change the **FCRYSTAL** parameter to alter t
Controlled Oscillator (VCO). To use an exter
to the external clock frequency and the MOD
CPU32Bug monitors the MODCLK* signal a
when calculating SCI baud rates.

Q: Why do certain baud rates fail to work after I
clock?

A: There is an integral relationship between the
rates, as per Section 5.6.3.1 SCI CONTROL
Manual, MC68332UM/AD (or in the previou
Manual, SIM32UM/AD), as defined by the fo

SCI baud = System

where SCBR equals {1, 2, 3, ..., 8191}. For a
between the Nominal Baud Rate and the Actu
be kept within 3% for reliable operation. Reli
the ability of the communications hardware a
device, such as found in the IBM-PC, might t

In summary, all baud rates may not be availa

Q: After I made the parameter change for an ext
on header P2 by jumping pin 28 to 64, nothin
signon message appears. Why doesn't it work

A: The trace between pins 2 and 3 of jumper J1
over pins 1-2 of J1 before the external clock s

Q: How can I change the number of data bytes i
command?

A: Change the **SRECMAX** parameter to alter th
efficient data S-records, but some loaders can
view/edit as text files. Thus the default is set

Q: How can I move the register module base to
MM bit in the Module Control Register (MC

A: Change the **MCR_AND** parameter so the MM
routine initializes the MCR register by first re
parameter value and then AND'ing the result
storing the resulting value back into the MCR

Q: How can I enable the Software Watchdog or
controlled by the write-once System Protectio

A: Change the **SYPCR_OR** and **SYPCR_AND**
placed into the SYPCR register. The **PWR_I**
first reading the register, OR'ing in the**SYPC**
result with the **SYPCR_AND** parameter valu
SYPCR register. As the Software Watchdog
values, some CPU32Bug commands may fail
be serviced, which causes a system reset. If th
message will never appear, as the MCU will